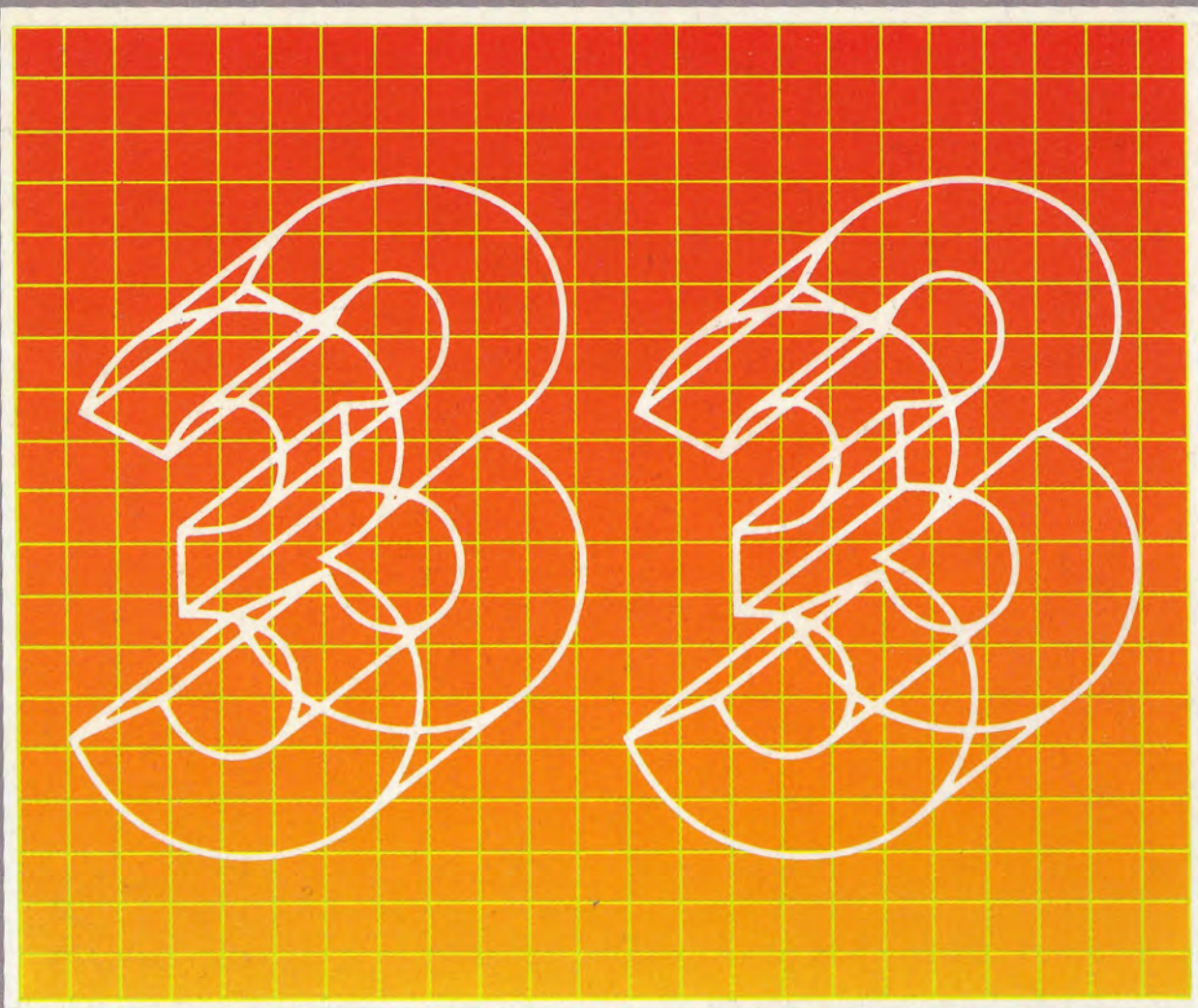


BIBLIOTECA PRACTICA

TALLER DE INFORMATICA

TRUCOS - SPRITES
BRICOLAGE DEL HARD
AULA ABIERTA
LOGO - TERMINOLOGIA

PROGRAMAS VALIDOS PARA AMSTRAD,
IBM, SPECTRUM, COMMODORE Y MSX



TRUCOS DE PROGRAMACION
Y DE DISEÑO DE GRAFICOS

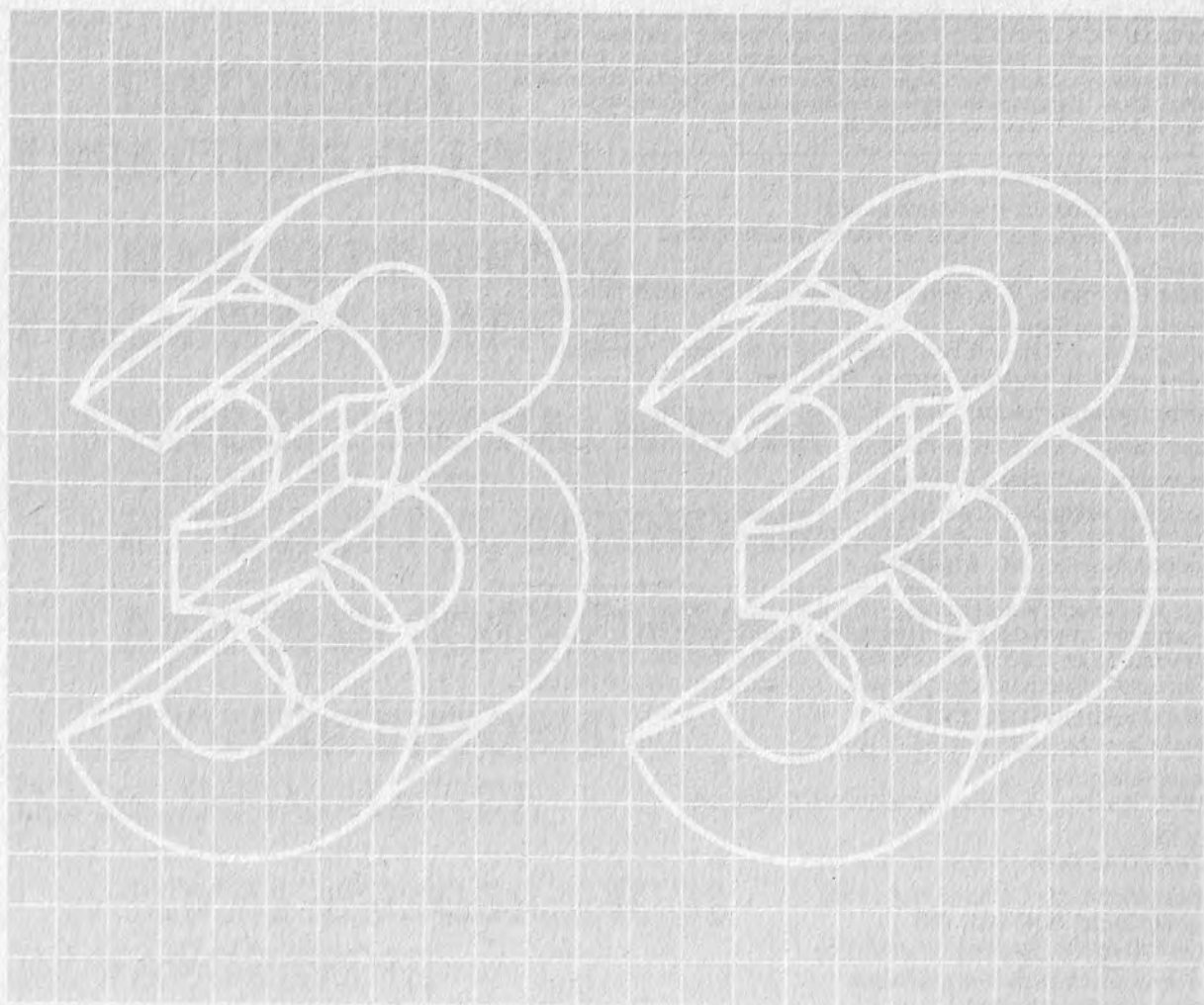
EDICIONES SIGLO CULTURAL

BIBLIOTECA PRACTICA

TALLER DE INFORMATICA

TRUCOS - SPRITES
BRICOLAGE DEL HARD
AULA ABIERTA
LOGO - TERMINOLOGIA

PROGRAMAS VALIDOS PARA AMSTRAD,
IBM, SPECTRUM, COMMODORE Y MSX



EDICIONES SIGLO CULTURAL

Una publicación de

EDICIONES SIGLO CULTURAL, S. A.

Director-editor:

RICARDO ESPAÑOL CRESPO.

Gerente:

ANTONIO G. CUERPO.

Directora de producción:

MARIA LUISA SUAREZ PEREZ.

Director de la colección:

JOSE ARTECHE, Ingeniero de Telecomunicación

Diseño:

BRAVO-LOFISH.

Maquetación:

D. S. M.

Dibujos:

JOSE OCHOA

Tomo 33. «Experiencias y prácticas en logo», Carlos Albert, Técnico de Informática; Adoración Llena Jubelo, Técnica de Informática. «Manejo de sprites y elementos gráficos», «Trucos y rutinas básicas», Francisco Morales. Técnico de Informática. «Aprende con el ordenador», AULA DE INFORMÁTICA APLICADA: Francisco Javier Gavilán, Diplomado en Telecomunicación; Alejandro Marcos, Licenciado en Química. «El Taller del Hardware», Carlos Rey, Ingeniero Industrial. «Pequeña historia de la Informática», «Temas monográficos de vanguardia», «Terminología», Blanca Arbizu, Técnico de Informática.

Ediciones Siglo Cultural, S. A.

Dirección, redacción y administración:

Pedro Teixeira, 8-2.º. Teléf. 455 09 99. 28020 Madrid.

Publicidad:

Gofar Publicidad, S. A. Benito de Castro, 12 bis. 28028 Madrid.

Distribución en España:

COEDIS, S. A. Valencia, 245. Teléf. 215 70 97. 08007 Barcelona.

Delegación en Madrid: Serrano, 165. Teléf. 411 11 48.

Distribución en Ecuador: Muñoz Hnos.

Distribución en Perú: DISELPESA.

Distribución en Chile: Alfa Ltda.

Importador exclusivo Cono Sur:

CADE, S.R.L. Pasaje Sud América, 1532. Teléf.: 21 24 64.

Buenos Aires - 1.290. Argentina.

Todos los derechos reservados. Este libro no puede ser, en parte o totalmente, reproducido, memorizado en sistemas de archivo, o transmitido en cualquier forma o medio, electrónico, mecánico, fotocopia o cualquier otro, sin la previa autorización del editor.

ISBN del tomo: 84-7688-172-X

ISBN de la obra: 84-7688-047-2

Fotocomposición:

ARTECOMP, S. A. Albarracín, 50. 28037 Madrid.

Imprime:

GRAFICAS EMA. (Madrid)

© Ediciones Siglo Cultural, S. A., 1986

Depósito legal: M-43.185-1986

Printed in Spain - Impreso en España.

Suscripciones y números atrasados:

Ediciones Siglo Cultural, S. A.

Pedro Teixeira, 8-2.º. Teléf.: 259 73 31. 28020 Madrid.

Noviembre, 1987

P.V.P. Canarias: 355.—Ptas.

INDICE

■ EXPERIENCIA Y PRACTICAS EN LOGO	5
■ MANEJO DE SPRITES Y ELEMENTOS GRAFICOS	14
■ TRUCOS Y RUTINAS BASICAS	25
■ EL TALLER DEL HARDWARE	31
■ APRENDER CON EL ORDENADOR	39
■ PEQUEÑA HISTORIA DE LA INFORMATICA	47
■ TEMAS MONOGRAFICOS DE VANGUARDIA	51
■ TERMINOLOGIA	57

EXPERIENCIA Y PRACTICAS EN LOGO

Listas de propiedades



AMOS a ver a continuación tres primitivas que nos permiten consultar las listas de propiedades asociadas a las variables y a los procedimientos.

— IMPROP nombre
significa IMprime PROPIe-

dades

Esta primitiva presenta en pantalla la lista de propiedades del nombre especificado.

El nombre puede ser una variable, un procedimiento o un grupo.

Veamos algunos ejemplos:

Primero asociamos a una variable una lista de propiedades.

```
? HAZ "A CASA
```

```
? PONPROP :A "BALCON "EXTERIOR
```

```
? PONPROP :A "TELEFONO "ROJO
```

Comprobamos que la variable A tiene asociadas estas propiedades.

```
? IMPROP :A
```

```
PONPROP :A "BALCON "EXTERIOR
```

```
PONPROP :A "TELEFONO "ROJO
```

En la pantalla han aparecido las dos propiedades con sus respectivos valores.

Si borramos de la variable A la propiedad BALCON.

```
? BOPROP :A "BALCON
```

Al consultar de nuevo la lista obtenemos:

```
? IMPROP :A
```

```
PONPROP :A "TELEFONO "ROJO
```

También podemos borrar la otra propiedad que tiene asociada la variable A.

```
? BOPROP :A "TELEFONO
```

Si intentamos consultar la lista de propiedades asociada a la variable A, en la pantalla no aparecerá nada.

```
? IMPROP :A
```

```
— IMPROPS
```

Esta otra primitiva presenta en la pantalla las listas de propiedades de todo lo que se encuentre en el espacio de trabajo.

Volvamos a asociar a la variable A las propiedades anteriores.

```
? PONPROP :A "BALCON "EXTERIOR
```

```
? PONPROP :A "TELEFONO "ROJO
```

Y asociemos unas cuantas propiedades al siguiente procedimiento:

```
? PARA "A1
```

```
> HAZ "A [ME ENCANTAN LAS ZANAHORIAS]
```

```
> ES :A
```

```
> FIN
```

```
? PONPROP "A1 "CAMPO "HORTALIZAS
```

```
? PONPROP "A1 "COCINA "ZANAHORIAS
```

Probemos ahora la primitiva IMPROPS:

```
? IMPROPS
```

```
PONPROP :A "BALCON "EXTERIOR
```

```
PONPROP :A "TELEFONO "ROJO
```

```
PONPROP "A1 "CAMPO "HORTALIZAS
```

```
PONPROP "A1 "COCINA "ZANAHORIAS
```

La relación de listas tanto de la variable A del procedimiento A1, ha aparecido en la pantalla.

Amplíemos el ejemplo trabajando con grupos.

Imagina que tenemos definidos en la memoria los siguientes procedimientos.

Podemos asociar listas de primitivas a los grupos que hayamos formado.

EXPERIENCIA Y PRACTICAS EN LOGO

RAIZ
TRONCO
RAMA
HOJA

Formemos un grupo con ellos:

? AGRUPA "ARBOL [RAIZ TRONCO
RAMA HOJA]

Y asociemos a este grupo una serie de propiedades:

? PONPROP "ARBOL "NOMBRE "PINO
? PONPROP "ARBOL "TAMAÑO "GRAN-
DE
? PONPROP "ARBOL "EDAD "25

Ahora en el espacio de trabajo está también el grupo ARBOL con una serie de propiedades, comprobémoslo:

? IMPROPS
PONPROP :A "BALCON "EXTERIOR
PONPROP :A "TELEFONO "ROJO
PONPROP :A1 "CAMPO "HORTALIZAS
PONPROP :A1 "COCINA "ZANAHORIAS
PONPROP "ARBOL "NOMBRE "PINO
PONPROP "ARBOL "TAMAÑO "GRAN-
DE
PONPROP "ARBOL "EDAD "25

— LPROP nombre

significa Lista de PROPIedades

Devuelve la lista de propiedades asociadas al nombre especificado.

La principal diferencia entre esta primitiva y la primitiva IMPROP es la forma de la lista devuelta; otra diferencia es que si utilizamos LPROP tenemos que indicar qué debe hacer el LOGO con la lista devuelta.

Vamos a consultar con esta primitiva las listas asociadas anteriormente a la variable A, al procedimiento A1 y al grupo ARBOL.

? ES LPROP :A
TELEFONO ROJO BALCON EXTERIOR

? ES LPROP "A1
COCINA ZANAHORIA CAMPO HORTALIZAS

? ES LPROP "ARBOL
EDAD 25 TAMAÑO GRANDE NOMBRE
PINO

Observa que el orden de aparición es el contrario al orden de asociación.

Veamos un último ejemplo con las tres primitivas:

? HAZ "A "TELEVISION
? PONPROP :A "NOTICIAS "TELEDIARIO
? PONPROP :A "CONCURSO [UN DOS
TRES]
? PONPROP :A "PUBLICIDAD [ATUN
CALVO]
? PONPROP :A "MUSICA [A TOPE]
? PONPROP :A "INFANTIL [BARRIO SESAMO]

? IMPROP :A
PONPROP :A "NOTICIAS "TELEDIARIO
PONPROP :A "CONCURSO [UN DOS
TRES]
PONPROP :A "PUBLICIDAD [ATUN
CALVO]
PONPROP :A "MUSICA [A TOPE]
PONPROP :A "INFANTIL [BARRIO SESAMO]

? IMPROPS
PONPROP :A "NOTICIAS "TELEDIARIO
PONPROP :A "CONCURSO [UN DOS
TRES]
PONPROP :A "PUBLICIDAD [ATUN
CALVO]
PONPROP :A "MUSICA [A TOPE]
PONPROP :A "INFANTIL [BARRIO SESAMO]

? LPROP :A
INFANTIL [BARRIO SESAMO] MUSICA
[A TOPE] PUBLICIDAD [ATUN
CLARO] CONCURSO [UN DOS
TRES] NOTICIAS TELEDIARIO

Cuadro resumen

— IMPROP nombre

Presenta las propiedades que están asociadas al nombre especificado.

— IMPROPS

Presenta las listas de propiedades de todo lo que se encuentra en el espacio de trabajo.

Si una variable no tiene valor, es imposible asociarle una lista de propiedades.

— LPROP nombre

Devuelve la lista de propiedades asociadas al nombre especificado.

Ejercicios

1. Dibuja una bombilla y establece un menú para seleccionar una de las siguientes opciones:

ENCENDER
APAGAR
TERMINAR

Prepara los procedimientos necesarios para:

- Rellenar la bombilla de color si se elige ENCENDER.
- Rellenar la bombilla del color de fondo si se elige APAGAR.
- Parar si se elige TERMINAR.

2. Realiza el siguiente dibujo.

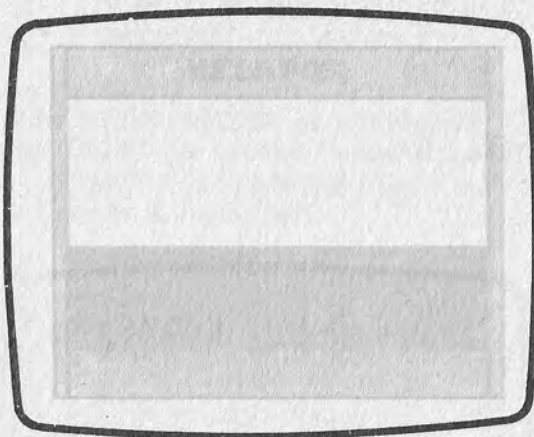


Fig. 1.

3. Define los procedimientos necesarios para que introduciendo el nombre de una provincia española, obtengamos las correspondientes siglas de la matrícula de sus vehículos.

Haz los cambios necesarios para realizar este proceso a la inversa, es decir, que introduciendo una matrícula obtengamos la provincia a la que corresponde.

4. Define los procedimientos necesarios para realizar el siguiente proceso:

1. Se pedirá que se introduzca un número de columna y de fila.

2. Se pedirá que se introduzca una palabra del número de caracteres que se desee.

3. Después de que una tecla haya sido pulsada, se borrará la pantalla y en ella ha de aparecer la palabra introducida a partir de la columna y fila que se han introducido. A continuación la palabra se enmarcará de la siguiente forma:

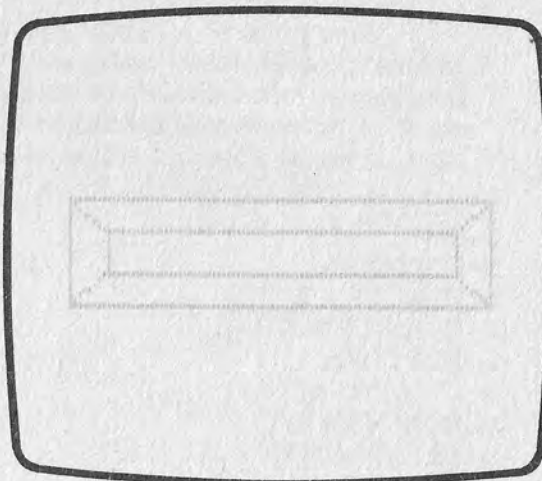


Fig. 2.

Este proceso tiene que ser válido para una palabra de cualquier longitud y situada en cualquier posición de la pantalla.

5. ¿Son correctas las siguientes órdenes?

- ? HAZ "A "ANIMALES
- ? PONPROP :A "PECES "TRUCHA
- ? PONPROP :A "MAMIFEROS "GATO
- ? ES LPROP
- ? HAZ "A "ANIMALES
- ? PONPROP :A "PECES "TRUCHA
- ? PONPROP :A "MAMIFEROS "GATO
- ? ES LPROP :A "PECES
- ? HAZ "A "ANIMALES
- ? PONPROP :A "PECES "TRUCHA
- ? PONPROP :A "MAMIFEROS "GATO
- ? IMPROP :A
- ? HAZ "A AZAR LL
- ? HAZ "X 10
- ? HAZ "Y 20
- ? PONPOS LISTA :A :B

Es posible asociarle a un procedimiento que no esté definido una lista de propiedades.

EXPERIENCIA Y PRACTICAS EN LOGO

■ Solución a los ejercicios

1

El procedimiento D inicializa la pantalla, ejecuta el procedimiento que dibuja la bombilla y muestra el siguiente menú:

```
ENCENDER = 1
APAGAR = 2
TERMINAR = 3
```

Si se pulsa el 1, la bombilla se rellena de color; si se pulsa el 2, la bombilla se rellena del color del fondo; si se pulsa el 3, la ejecución termina, si no, se vuelve a empezar.

? PARA D

```
> PM
> FORMAN
> PONCURSOR [1 2]
> ES [ENCENDER = 1]
> PONCURSOR [1 4]
> ES [APAGAR = 2]
> PONCURSOR [1 6]
> ES [TERMINAR = 3]
> SL
> PONPOS [0 10]
> HAZ "A LC
> BL
> SI :A = 1 [PONCL 2 RELLENA]
> SI :A = 2 [PONCL 0 RELLENA]
> SI :A = 3 [ALTO] [D]
> FIN
```

El procedimiento FORMAM dibuja la bombilla.

? PARA FORMAM

```
> SL
> PONCL 1
> PONPOS [-15 -20]
> BL
> PONRUMBO 315
> AV 5
> GD 45
> AV 15
> GI 45
> AV 5
> GD 45
```

```
> AV 5
> GI 45
> AV 10
> REPITE 28 [AV 6 GD 10]
> PONRUMBO 225
> AV 10
> PONRUMBO 270
> AV 30 RE 30
> PONRUMBO 180
> AV 5
> GD 90
> AV 30 RE 30
> GI 45
> AV 5
> GI 45
> AV 15
> GD 45
> AV 5
> GD 45
> AV 15
> PONRUMBO 0
> SL
> PONPOS [-20 -17]
> BL
> REPITE 4 [GD 85 AV 26
  GI 170 AV 26 PONRUMBO 0]
> FIN
```

Al ejecutar el procedimiento D, en la pantalla aparece el menú y la bombilla apagada.

? D

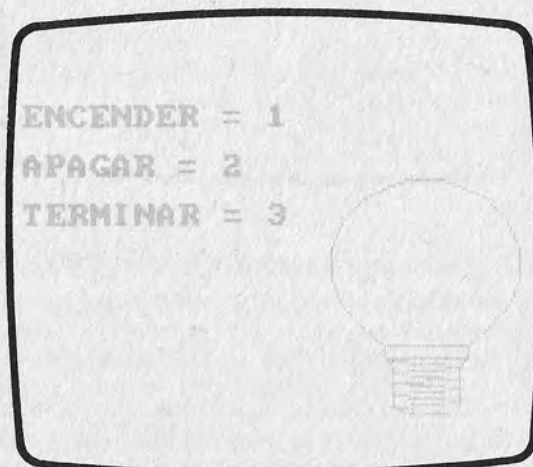


Fig. 3.

Para consultar las listas de propiedades de una variable o de un procedimiento podemos utilizar la primitiva IMPROP.

Si ahora pulsamos el 1, la bombilla se enciende.

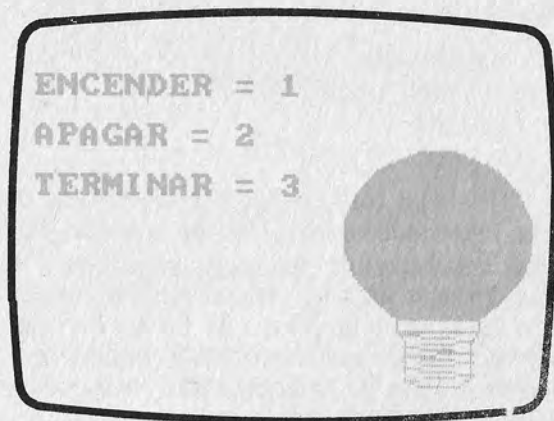


Fig. 4.

2

Definimos un procedimiento para inicializar la pantalla y para ejecutar el procedimiento que dibuja el puesto de helados.

```
? PARA D1
> BP
> PUESTO
> PONCURSOR [16 3]
> ES "HELADOS"
> FIN
```

PUESTO es el procedimiento que dibuja el puesto de helados.

```
? PARA PUESTO
> SL
> PONCL 3
> PONPOS [-80 -40]
> BL
> REPITE 2 [AV 140 GD 90
  AV 160 GD 90]
> SL
> PONPOS [-75 -40]
> BL
> AV 140
> SL
> PONPOS [-78 0]
> PONCL 1
> BL RELLENA
> PONCL 3
> SL
```

```
> PONPOS [75 -40]
> BL
> AV 140
> SL
> PONPOS [78 0]
> PONCL 1
> BL RELLENA
> SL
> PONPOS [-75 -20]
> BL
> PONCL 3
> GD 90 AV 150
> SL
> PONPOS [0 -30]
> PONCL 1
> BL RELLENA
> SL
> PONPOS [-75 20]
> BL PONCL 3
> AV 150
> SL
> PONPOS [0 -10]
> BL RELLENA
> SL
> PONPOS [-75 80]
> BL
> AV 150
> SL
> PONPOS [0 90]
> PONCL 1
> BL RELLENA
> FIN
```

Consigue el dibujo tecleando:

? D1

3

En primer lugar, creamos dos listas, una con los nombres de las provincias y otra con las matrículas de cada una de estas provincias. Los elementos de cada lista tienen que estar ordenados, es decir, tienen que corresponderse.

Esto lo realiza el procedimiento LISTAS:

```
? PARA LISTAS
> HAZ "A [AB A AL AV
  B BA BI BU CC CA CE
  CS CR CO C CU GC GE
  GR GU H HU J LE L LO
```

Si utilizamos la primitiva LPROP, debemos indicarle al Logo qué debe hacer con las listas de propiedades que devuelva esta primitiva.

EXPERIENCIA Y PRACTICAS EN LOGO

```
LU M MA ML MU NA OR
O P PM PO SA SS S SG
SE SO T TF TE TO V VI
VA ZA Z AND GBZ]
> HAZ "B [ALBACETE ALICANTE
ALMERIA AVILA BARCELONA
BADAJOZ VIZCAYA BURGOS
CACERES CADIZ CEUTA
CASTELLON CIUDADREAL
CORDOBA CORUÑA CUENCA
GRANCANARIA GERONA GRANA-
DA
GUADALAJARA HUELVA HUESCA
JAEN LEON LERIDA LOGROÑO
LUGO MADRID MALAGA MELILLA
MURCIA NAVARRA ORENSE
ASTURIAS PALENCIA BALEARES
PONTEVEDRA SALAMANCA GUI-
PUZCOA
SANTANDER SEGOVIA SEVILLA
SORIA TARRAGONA TENERIFE
TERUEL TOLEDO VALENCIA
ALAVA VALLADOLID ZAMORA
ZARAGOZA ANDORRA GIBRAL-
TAR]
> HAZ "I I
> FIN
```

En la variable A, guardamos la lista de las matrículas y en la variable B la lista de las provincias.

El procedimiento P1 es el que presenta la pantalla en donde se indica la provincia, la matrícula y el mensaje de petición del nombre de una provincia:

```
? PARA P1
> PONCURSOR [10 5]
> ES "PROVINCIA
> PONCURSOR [23 5]
> ES "MATRICULA
> SL
> PONPOS [-90 40]
> BL
> REPITE 2 [AV 40 GD 90
AV 196 GD 90]
> AV 20 GD 90
> AV 196 RE 98
> GI 90 AV 20
> RE 40
> PONCURSOR [10 12]
> ES [INTRODUCE LA PROVINCIA]
> FIN
```

P2 realiza una lectura desde el teclado de una lista en la posición adecuada. La lista introducida se le asigna a la variable A:

```
? PARA P2
> PONCURSOR [10 7]
> HAZ "L LL
> FIN
```

Una vez que se ha introducido la provincia, leemos los elementos de la lista de provincias con la ayuda del contador I. Cada elemento de esta lista lo asignamos a la variable P y lo transformamos en una lista. Cuando el elemento que se está leyendo coincide con la variable L (que es la que contiene la provincia introducida), ordenamos que se escriba el elemento que marca el contador I de la lista de matrículas:

```
? PARA P3
> HAZ "P (ELEMENTO :I :B)
> HAZ "P LISTA :P
> SI :P = :L [PONCURSOR
[23 7] ES ELEMENTO :I :A]
[HAZ "I :I + 1 P3]
> FIN
```

Por último, el procedimiento P5 es el que nos da opción a introducir otra provincia:

```
? PARA P5
> PONCURSOR [10 12]
> ES [¿OTRA PROVINCIA? S / N]
> HAZ "T LC
> SI :T = "S [TO]
> SI :T = "N [ALTO]
> P5
> FIN
```

Una vez que tenemos todos los procedimientos necesarios para realizar el proceso, los englobamos en el procedimiento TO:

```
? PARA TO
> BP
> BT
> OT
> LISTAS
> P1
> P2
> P3
> P5
> FIN
```

Si lo ejecutamos:

Podemos transformar una palabra en una lista.

? TO

en la pantalla aparece:



Fig. 5.

Si ahora introducimos el nombre de una provincia, por ejemplo, SALAMANCA, la pantalla aparece de la siguiente forma:

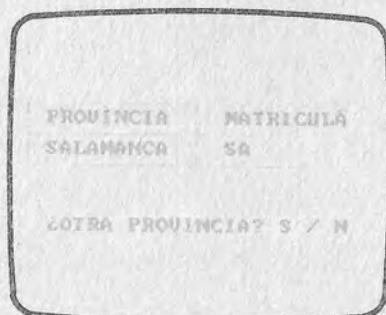


Fig. 6.

Si el nombre de la provincia es compuesto, por ejemplo, GRAN CANARIA, tienes que escribirlo junto:

Los cambios que hay que hacer para realizar el proceso a la inversa son:

En el procedimiento P1 intercambiamos los nombres de provincia y matrícula:

```
? PARA P1
> PONCURSOR [10 5]
> ES "MATRICULA"
> PONCURSOR [23 5]
> ES "PROVINCIA"
> SL
> PONPOS [-90 40]
> BL
> REPITE 2 [AV 40 GD 90
> AV 196 GD 90]
> AV 20 GD 90
> AV 196 RE 98
> GI 90 AV 20
> RE 40
> PONCURSOR [10 12]
```

```
> ES [INTRODUCE LA MATRICULA]
> FIN
```

En el P3 cambiamos las variables que contienen las listas. Vamos leyendo de la lista de matrículas y una vez encontrada escribimos el elemento correspondiente de la lista de provincias:

```
? PARA P3
> HAZ "P ( ELEMENTO :I :A )
> HAZ "P LISTA :P
> SI :P = :L [PONCURSOR [22 7]
ES ELEMENTO :I :B]
[HAZ "I :I + 1 P3]
> FIN
```

En P5 lo único que tenemos que hacer es cambiar el mensaje:

```
? PARA P5
> PONCURSOR [10 12]
> ES [¿OTRA MATRICULA? S / N]
> HAZ "T LC
> SI :T = "S [TO]
> SI :T = "N [ALTO]
> P5
> FIN
```

4.

Definimos el procedimiento A para que en la pantalla aparezcan los mensajes de petición de la columna, la fila y la palabra que se desea que aparezca:

```
? PARA A
> PONCURSOR [2 2]
> ES [INTRODUCE LA COLUMNA]
> PONCURSOR [23 2]
> HAZ "C LL
> HAZ "C UL :C
> PONCURSOR [2 5]
> ES [INTRODUCE LA FILA]
> PONCURSOR [20 5]
> HAZ "F LL
> HAZ "F UL :F
> PONCURSOR [2 8]
> ES [INTRODUCE LA PALABRA]
> PONCURSOR [23 8]
> HAZ "A LL
> HAZ "A UL :A
> TECLA
> FIN
```

Podemos operar con el resultado que nos devuelve la primitiva CUENTA.

EXPERIENCIA Y PRACTICAS EN LOGO

En la variable C guardamos el número de la columna y en la variable F el de la fila. La palabra que introducimos la asignamos a la variable A.

Al final de este procedimiento hacemos una llamada a TECLA para pasar a la pantalla siguiente en vez que se haya pulsado una tecla. El procedimiento TECLA es el siguiente:

```
? PARA TECLA
> PONCURSOR [12 22]
> ES [PULSA UNA TECLA]
> HAZ "T LC
> BP
> BT
> FIN
```

Una vez que se pulse una tecla, en la pantalla tiene que aparecer la palabra introducida a partir de la columna y la fila que se ha introducido. Esto lo conseguimos con el procedimiento B:

```
? PARA B
> PONCURSOR LISTA :C :F
> ES :A
> FIN
```

Una vez que la palabra se ha escrito en la pantalla, situamos a la Tortuga en el principio de la primera letra y en su parte inferior.

La posición en X la calculamos multiplicando el valor de la columna donde se ha empezado a escribir la palabra por ocho, que son los puntos que ocupan un carácter en horizontal, y restándole 160, que es el punto extremo izquierdo de la pantalla.

La posición en Y la calculamos multiplicando la fila por diez, que es lo que ocupa un carácter en vertical si dejamos un punto en blanco por arriba y por debajo de él, y restándole 116, que es el punto superior de la pantalla.

Una vez situada la Tortuga en la posición adecuada, calculamos sobre la variable CU la longitud horizontal en puntos que tiene la palabra. Para realizar esto, contamos los elementos que tiene la palabra (longitud en caracteres). El número de elementos lo multiplicamos por ocho, que es lo que ocupa cada uno. De esta forma obtenemos el total de puntos en horizontal que ocupa.

Empezamos a dibujar el marco, sabiendo que la longitud de la palabra en horizontal está en la variable CU y que su longitud vertical es de diez puntos.

El procedimiento C es el que realiza todas estas operaciones y dibuja el marco:

```
? PARA C
> SL
> PONPOS LISTA -160+*C*8 116-( :F * 10)
> BL
> HAZ "CU CUENTA :A
> HAZ "CU :CU * 8
> AV 10 GD 90
> AV :CU GD 90
> AV 10 GD 90
> AV :CU
> PONRUMBO 0
> GD 45 RE 10
> GI 45 AV 25
> GI 45 RE 10
> AV 10 GD 135
> AV :CU + 15
> GI 45 RE 10
> AV 10 GD 135
> AV 25 GI 45
> RE 10 AV 10
> GD 135
> AV :CU + 15
> FIN
```

Una vez que tenemos todos los procedimientos necesarios, definimos TO, que es el que inicializa el proceso y engloba a todos los anteriores:

```
? PARA TO
> BP
> BT
> OT
> A
> B
> C
> PONCURSOR [0 22]
> FIN
```

Si lo ejecutamos:

```
? TO
```

e introducimos los siguientes datos:

```
10
```

PONPOS admite variables siempre que se indique con la primitiva LISTA.

11

ORDENADOR

en la pantalla aparecerá:

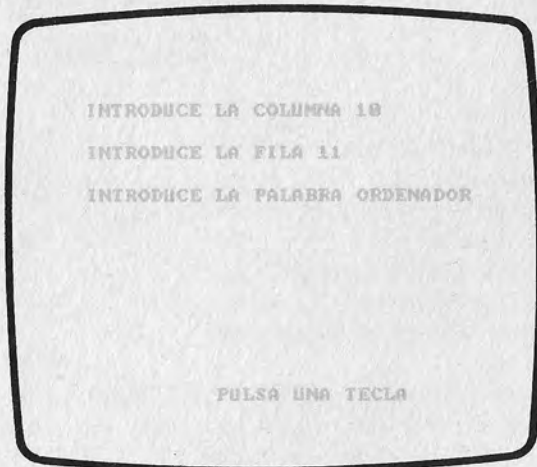


Fig. 7.

una vez que pulsemos una tecla en la pantalla aparecerá:

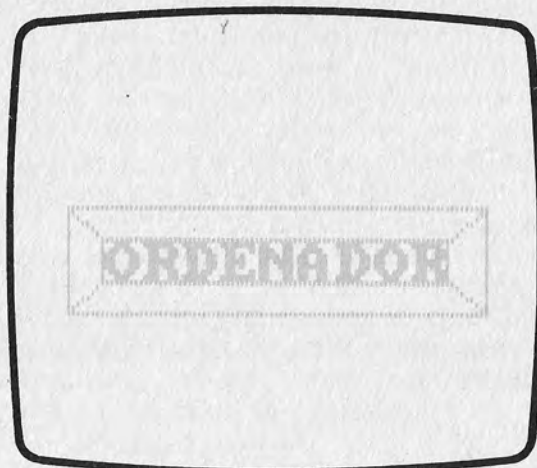


Fig. 8.

5.

```
— ? HAZ "A "ANIMALES
? PONPROP :A "PECES "TRUCHA
? PONPROP :A "MAMIFEROS "GATO
INCORRECTO
```

El Logo devuelve el mensaje:

FALTAN DATOS PARA LPROP

No hemos indicado la variable o el procedimiento que queremos consultar con la primitiva LPROP.

```
— ? HAZ "A "ANIMALES
? PONPROP :A "PECES "TRUCHA
? PONPROP :A "MAMIFEROS "GATO
? ES LPROP :A "PECES
```

INCORRECTO.

El LOGO devuelve el mensaje de error:

NO SE QUE DEBO HACER CON PECES

La primitiva LPROP no precisa de ninguna propiedad específica.

```
— ? HAZ "A "ANIMALES
? PONPROP :A "PECES "TRUCHA
? PONPROP :A "MAMIFEROS "GATO
? IMPROP :A
```

CORRECTO.

En la pantalla aparece:

```
PONPROP "ANIMALES "MAMIFEROS
"GATO
PONPROP "ANIMALES "PECES "TRU-
CHA
```

```
— ? HAZ "A AZAR LL
```

INCORRECTO.

Si, por ejemplo, introducimos la lista:

DONDE VAS

recibiremos el mensaje:

AZAR NO ADMITE [DONDE VAS]
COMO DATO

```
— ? HAZ "X 10
? HAZ "Y 20
? PONPOS LISTA :A :B
```

CORRECTO.

La primitiva PONPOS admite datos variables siempre que se indique la primitiva LISTA.

Las listas admiten gran cantidad de elementos.

Una lista de un solo elemento podemos transformarla en una palabra.

MANEJO DE SPRITES Y ELEMENTOS GRAFICOS

La alta resolución en el AMSTRAD



OSIBLEMENTE el AMSTRAD sea el ordenador con más modos de pantalla en alta resolución del mercado. También podemos decir que es el ordenador con mayor capacidad gráfica de todos los que tratamos en esta

colección.

El AMSTRAD dispone de tres modos de pantalla por todos conocido (MODE 0, MODE 1 y MODE 2), en cada uno de ellos la resolución es distinta y el número de colores que podemos utilizar también es distinto. Cada modo de pantalla puede servirnos para un propósito en particular. El modo 0 nos puede servir para realizar juegos con gran colorido. El modo 1 puede servirnos para cualquier programa que queramos hacer y que requiera una discreta distribución del color. Por otro lado, el modo 2 está muy enfocado a realizar programas en los cuales necesitamos mucha información simultánea en la pantalla.

Al igual que en cada modo de pantalla tenemos una serie de caracteres por línea que varía entre 20 y 80, en cada modo tenemos una resolución distinta y un número de colores que podemos utilizar. La figura 1 nos muestra una tabla en la cual se ve la realización entre los caracteres por línea, la resolución y los colores que podemos utilizar.

MODOS	RESOLUCIÓN	CARACTERES POR LÍNEA	COLORES
MODE 0	640x200	80x5 líneas	16
MODE 1	320x200	40x5 líneas	4
MODE 2	640x200	80x5 líneas	2

Fig. 1.—Los modos de pantalla y su resolución en el AMSTRAD.

El tener cada tipo de pantalla una resolución distinta y ocupar el mismo espacio visual, implica que el tamaño de los caracteres y de los puntos son distintos de un modo a otro. Las figuras 2, 3 y 4 nos muestran la distribución de dichos puntos.

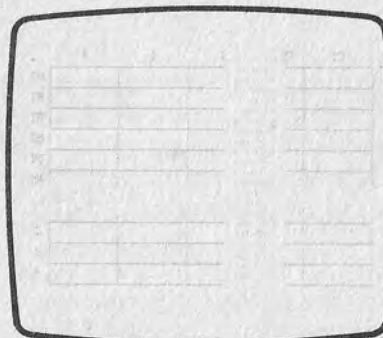


Fig. 2.—Modo 0 de pantalla.

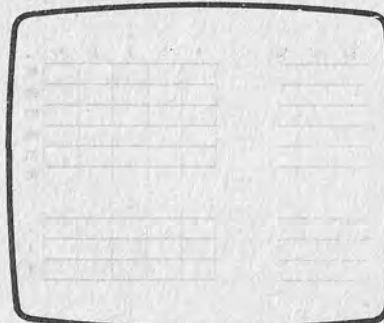


Fig. 3.—Modo 1 de pantalla. Sus puntos son la mitad que en el modo 0.

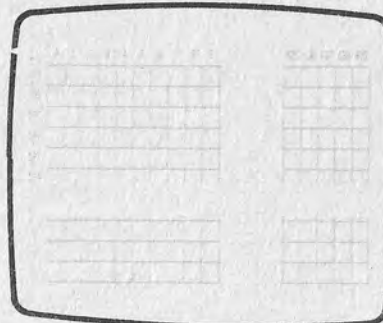


Fig. 4.—Modo 2 de pantalla. Sus puntos son la mitad que en el modo 1.

Como se ve en estas figuras, la resolución vertical no varía en ninguno de los tres modos de pantalla.

MANEJO DE SPRITES Y ELEMENTOS GRAFICOS

■ Los comandos de la alta resolución en el AMSTRAD

A continuación vamos a hacer un estudio de los comandos e instrucciones de que dispone el AMSTRAD para dibujar líneas, puntos, etc., en alta resolución. Todo lo que se diga servirá para los tres modos de pantalla, aunque, por comodidad, siempre hablaremos, del modo 1 y los programas los realizaremos en la resolución de 320×200 .

■ Las instrucciones PLOT y PLOTR

Lo primero que se nos plantea es cómo dibujar un simple punto en nuestra pantalla.

Para ello contamos la instrucción PLOT. Su sintaxis es la siguiente:

PLOT X,Y,C

donde X e Y son las coordenadas del punto de la pantalla que queremos iluminar y C es el color que queremos que tenga el punto. Este comando es muy sencillo y no tiene ninguna complicación. La diferencia entre un modo y otro de pantalla será en el tamaño del punto. Por supuesto, si estamos en el modo 0 y hacemos un PLOT en 300, 100, como dicha coordenada se encuentra fuera de los límites de la pantalla, no lo veremos.

```
1000 REM *****
1010 REM * NOCHE ESTRELLADA *
1020 REM *****
1030 REM
1040 REM *****
1050 REM * PROGRAMA DEMOSTRACION PARA VER EL *
1060 REM * USO DEL PLOR EN EL AMSTRAD.      *
1070 REM *****
1080 REM
1090 MODE 1
1100 CLS
1110 DIM X%(500)
1120 DIM Y%(500)
1130 REM
1140 REM *** DIBUJO DEL CIELO ESTRELLADO ***
1150 REM
1160 LOCATE 1,25
1170 PRINT "DIBUJANDO EL CIELO";
1180 RANDOMIZE TIME
1190 FOR I=1 TO 500
1200   LET X=INT(RND(1)*320)
1210   LET Y=INT(RND(1)*200)
1220   LET C=INT(RND(1)*3+1)
1230   PLOT X,Y,C
1240   LET X%(I)=X
1250   LET Y%(I)=Y
1260 NEXT I
1270 REM
1280 REM *** BUCLE DE RETARDO ***
1290 REM
1300 FOR I=1 TO 1500
1310 NEXT I
1320 REM
1330 REM *** BORRADO DEL CIELO ESTRELLADO ***
1340 REM
1350 PRINT CHR$(7);CHR$(7);
1360 LOCATE 1,25
1370 PRINT "BORRANDO EL CIELO ";
1380 FOR I=1 TO 500
1390   LET X=X%(I)
1400   LET Y=Y%(I)
1410   PLOT X,Y,0
1420 NEXT I
1430 LOCATE 1,1,0
```

El programa que aparece a continuación nos muestra cómo se dibujan puntos con el comando PLOT.

Una mirada atenta a la segunda parte del programa nos permitirá ver que si queremos borrar un punto no hace falta nada más que decir que el color de la tinta a utilizar sea igual que el del papel de fondo.



Fig. 5.—Ejecución del programa 1.

Existe otra instrucción que, se puede decir, es una variante de PLOT. Su sintaxis es:

PLOTR, X,Y,C

La R viene de RELATIVE (RELATIVO) y sirve para dibujar un punto en una posición de

la pantalla X,Y en color C relativa a la que nos encontramos. Pongamos un ejemplo. Vamos a suponer que hemos iluminado el punto que se encuentra en 100,100. Si queremos iluminar el que se encuentra en 110,100, podemos escribir:

PLOT 110,110,1

pero eso implica que necesitamos conocer el punto sobre el que se encuentra el cursor gráfico. Hay veces en las cuales, no sabiendo en qué parte de la pantalla nos encontramos, sabemos que queremos dibujar un punto a cierta distancia del que acabamos de dibujar. Podemos necesitar iluminar un punto 10 pasos más a la derecha y 3 más abajo del que acabamos de dibujar. Para ello utilizamos la instrucción PLOTR. Con ella podemos dibujar un punto que se encuentra a cierta distancia del último iluminado. Si tomamos el ejemplo anterior, sólo tenemos que escribir:

PLOTR 10,-3,1

Veamos a continuación una variante del programa 1 que hace lo mismo, pero utilizando la instrucción PLOTR.

```

1000 REM *****
1010 REM * NOCHE ESTRELLADA *
1020 REM *****
1030 REM
1040 REM *****
1050 REM * PROGRAMA DEMOSTRACION PARA VER EL *
1060 REM * USO DEL PLOT EN EL AMSTRAD.      *
1070 REM *****
1080 REM
1090 MODE 1
1100 CLS
1110 DIM X%(500)
1120 DIM Y%(500)
1130 REM
1140 REM *** DIBUJO DEL CIELO ESTRELLADO ***
1150 REM
1160 LOCATE 1,25
1170 PRINT "DIBUJANDO EL CIELO";
1180 RANDOMIZE TIME
1190 LET XO=0
1200 LET YO=0
1210 FOR I=1 TO 500
1220   LET X=XO+INT(RND(1)*10)
1230   LET Y=YO+INT(RND(1)*10)
1240   LET C=INT(RND(1)*3+1)
1250   PLOT X,Y,C
1260   LET X%(I)=X-XO
1270   LET Y%(I)=Y-YO
1280   LET XO=X
1290   LET YO=Y
1300   IF XO+10>320 THEN LET XO=0

```

MANEJO DE SPRITES Y ELEMENTOS GRAFICOS

```
1310 IF YO+10>200 THEN LET YO=0
1320 NEXT I
1330 REM
1340 REM *** BUCLE DE RETARDO ***
1350 REM
1360 FOR I=1 TO 1500
1370 NEXT I
1380 REM
1390 REM *** BORRADO DEL CIELO ESTRELLADO ***
1400 REM
1410 LET XO=0
1420 LET YO=0
1430 PRINT CHR$(7);CHR$(7);
1440 LOCATE 1,25
1450 PRINT "BORRANDO EL CIELO ";
1460 FOR I=1 TO 500
1470 LET X=XO+X%(I)
1480 LET Y=YO+Y%(I)
1490 PLOT X,Y,0
1500 LET XO=X
1510 LET YO=Y
1520 IF XO+10>320 THEN LET XO=0
1530 IF YO+10>200 THEN LET YO=0
1540 NEXT I
1550 LOCATE 1,1
1560 END
```

Fíjese el lector en la parte del programa que se encarga de borrar los puntos anteriormente dibujados.

Las instrucciones MOVE y MOVER

El comando MOVE nos sirve para colocar el cursor gráfico en una cierta posición de la pantalla sin dibujar. Este comando sirve para que más tarde, cuando usemos la instrucción DRAW, definir cuál va a ser la coordenada de la pantalla de la cual partimos. Su sintaxis es la siguiente:

MOVE X,Y

donde X e Y son las coordenadas de la pantalla donde nos queremos mover.

La orden MOVER realiza la misma función que MOVE, pero el movimiento que realiza es relativo a donde nos encontramos. Por ello, si decimos MOVER 10,10, el cursor gráfico se moverá, siempre sin dibujar, diez puntos a la derecha y diez puntos hacia arriba.

Las instrucciones DRAW y DRAWR

Nos permiten dibujar una línea recta desde el lugar donde se encuentra el cursor gráfico hasta las coordenadas que nosotros le digamos. Su sintaxis es:

DRAW X,Y,C

donde X e Y son las coordenadas absolutas de la pantalla hasta donde queremos que llega la línea recta. C es el color que queremos que tenga dicha línea recta.

El origen de la línea lo podemos definir mediante un PLOT, PLOT, MOVE, MOVER o desde la última instrucción DRAW o DRAWR realizada. Una vez trazada la línea, el cursor gráfico queda en el último punto de la línea en espera de un nuevo comando de gráficos.

Como habrá adivinado el lector, la instrucción DRAWR dibuja, igualmente, líneas desde la posición actual del cursor hasta la otra posición de la pantalla cuyas coordenadas no sabemos, pero sí sabemos el desplazamiento que existe entre la antigua posición y la actual.

Para ilustrar un poco lo que acabamos de ver, vamos a introducir en nuestro AMS-TRAD el programa 3.

```

1000 REM *****
1010 REM * NEW YORK *
1020 REM *****
1030 REM
1040 REM *****
1050 REM * PROGRAMA DEMOSTRACION PARA VER EL USO DE *
1060 REM * LOS COMANDOS MOVE Y DRAW EN EL AMSTRAD. *
1070 REM *****
1080 REM
1090 REM *** INICIALIZACION Y DIBUJO DE LA PANTALLA ***
1100 REM
1110 MODE 1
1120 CLS
1130 LOCATE 1,25
1140 PRINT "VISTA AEREA DE NEW YORK";
1150 LOCATE 1,1
1160 PRINT CHR$(135);STRING$(38,131);CHR$(139)
1170 FOR I=2 TO 15
1180     PRINT CHR$(133);TAB(40);CHR$(138)
1190 NEXT I
1200 PRINT CHR$(141);STRING$(38,140);CHR$(142)
1210 LET XMIN=15
1220 LET YMIN=15
1230 LET XMAX=300
1240 REM
1250 REM *** IMPRESION DE MENSAJE PARA TERMINAR ***
1260 REM
1270 RANDOMIZE TIME
1280 LOCATE 1,20
1290 PRINT "PULSA 'P' PARA PARAR"
1300 REM
1310 REM *** DIBUJO DE NEW YORK ***
1320 REM
1330 LET X1=INT(RND(1)*XMAX+XMIN)
1340 LET Y1=INT(RND(1)*100+YMIN)
1350 LET X2=INT(RND(1)*XMAX+XMIN)
1360 LET Y2=120
1370 LET C=INT(RND(1)*3+1)
1380 IF ABS(X2-X1)>80 THEN GOTO 1330
1390 MOVE X1,Y1
1400 DRAW X2,Y1,C
1410 DRAW X2,Y2,C
1420 DRAW X1,Y2,C
1430 DRAW X1,Y1,C
1440 REM
1450 REM *** LECTURA DEL TECLADO Y RETARDO ***
1460 REM
1470 FOR I=1 TO 100
1480 IF INKEY$="P" THEN GOTO 1510
1490 NEXT I
1500 GOTO 1330
1510 CLS
1520 END

```

MANEJO DE SPRITES Y ELEMENTOS GRAFICOS

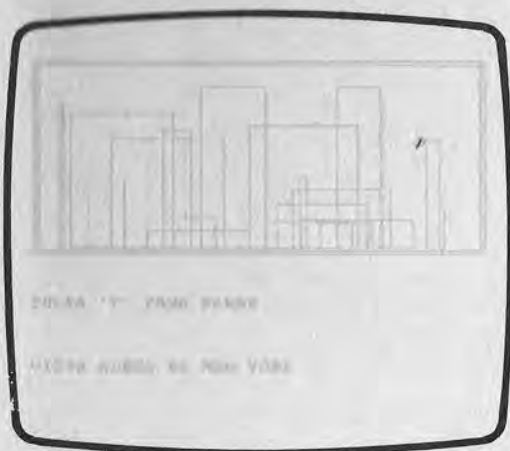


Fig. 6.—Ejecución del programa 3.

Las instrucciones WINDOW y ORIGIN

Antes de continuar viendo los comandos de que disponemos en alta resolución, vamos a ver dos nuevas instrucciones muy potentes. La primera de ellas, WINDOW, nos permitirá definir ventanas dentro de la pantalla. Una ventana es una parte de la pantalla que actúa de forma independiente al resto de la pantalla. Podemos hacer que un listado salga por una de las ventanas y no se meta por el resto de la pantalla.

Para definir una ventana tenemos que decirle al ordenador qué número de ventana va a ser la que vamos a crear y qué límites tendrá. Su sintaxis es:

`WINDOW #No. ventana,X1,X2,Y1,Y2`

El número de ventana es el número que asignamos a dicha parte de la pantalla y que será el que utilizemos para decirle al ordenador que realice operaciones sobre ella. X1 e Y1 son las coordenadas (en caracteres) de la esquina superior izquierda de la pantalla. X2 e Y2 son las coordenadas, también en caracteres, de la esquina inferior derecha de la ventana. Un ejemplo de definición de ventana podría ser:

`WINDOW #1,1,10,1,10`

con lo que tendríamos una ventana que se encontraría en la esquina superior izquierda de la pantalla y que tendría una anchura y una altura de 10 caracteres.

Una vez que hemos definido la ventana, para utilizarla no hay ningún problema, lo único que tenemos que hacer es especificar qué

ventana vamos a utilizar. Por ejemplo, si queremos imprimir la palabra HOLA, sólo tendremos que poner:

`PRINT #1,"HOLA"`

para que aparezca en esa pantalla. Si lo que queremos es borrar sólo el contenido de la misma, y no el resto de la pantalla, podemos escribir:

`CLS#1`

Podemos hacer que cambie el color del fondo de dicha ventana con sólo escribir:

`PAPER#1,2:CLS#1`

Como el lector puede apreciar, las ventanas funcionan como una pequeña pantalla dentro de la pantalla. No tenemos que preocuparnos por que los mensajes que escribamos no entren en el ancho de la venta, ya que el propio ordenador se encarga de escribir lo que no quepa en la siguiente línea.

Las instrucciones que podemos utilizar con las ventanas son las siguientes:

`CLS, INPUT, LINE INPUT, LIST, LOCATE, PAPER, PEN, POS, PRINT, TAG, TAFOFF, VPOS, WRITE.`

En todos ellos se coloca el número de la ventana al principio del comando, tal y como hemos visto con PRIN y con CLS, salvo en LIST, cuyo formato es:

`LIST No. línea inicial-No. línea final,#No. de ventana`

Por otro lado, veamos cuál es la función de la sentencia ORIGIN. Con ella podemos definir una ventana de gráficos, y no de caracteres, como hacemos con WINDOW, especificando incluso cuál va a ser su centro. Su sintaxis es:

`ORIGIN X,Y,X1,X2,Y1,Y2`

X e Y son las coordenadas que a partir de ese momento se tomarán como centro de las coordenadas. Por ejemplo, si escribimos en el ordenador:

`ORIGIN 100,100:PLOT 0,0`

lo que hará el ordenador será poner el origen de coordenadas en la posición 100,100 y, como ahora está es el origen, al escribir un punto en 0,0 se ilumina el que antes era 100,100.

X1 e Y1 son las coordenadas, en PIXELS, de la esquina superior izquierda de ventana que estamos definiendo.

X2 e Y2 son las coordenadas, en PIXELS, de la esquina inferior derecha de la ventana que estamos definiendo.

Los cuatro últimos argumentos de esta instrucción son totalmente opcionales. Si no los ponemos, el ordenador entiende que toda la pantalla es la ventana.

La instrucción CLG

Con ella podemos borrar los gráficos que se hayan dibujado en la pantalla, sin borrar ninguna letra del texto. Mientras CLS borra todo, letras y gráficos, este comando sólo borra los gráficos. Su sintaxis es:

CLG C

donde C es el color que queremos que tome la pantalla tras haberla borrado.

Las instrucciones TAG y TAGOFF

Muchas veces, al trabajar en alta resolución, tenemos que imprimir mensajes. Con el AMSTRAD no tenemos ningún problema, ya que con la sentencia LOCATE podemos colocar el cursor donde deseemos e imprimir lo que queramos. Pero hay veces que nos interesa escribir un mensaje o unos números en una posición intermedia entre dos caracteres o dos filas. Para ello, podemos utilizar la función TAG, que no necesita ningún parámetro. Si escribimos:

LOCATE 10,10:PRINT "HOLA"

el mensaje aparecerá en la columna 10 de la fila 10, pero si lo queremos imprimir en la columna 10.5 de la fila 10.5, tendremos que hacer uso de la sentencia TAG. Para escribir en esta posición, sólo tenemos que hacer lo siguiente:

TAG:PLOT 76,76:PRINT "HOLA"

La instrucción TAGOFF nos sirve para anular la función TAG.

Las funciones TEST y TESTR

Muchas veces, sobre todo al realizar gráficos en movimiento, nos interesa saber qué color tiene un determinado punto en un momento del programa. Para ello contamos con la instrucción TEST, a la cual le añadimos

las coordenadas del punto que queremos TESTear. El resultado será un número entre 0 y el número de colores permitido en el tipo de pantalla en que nos encontramos.

A=TEST X,Y

Por supuesto, TESTR realiza la misma función, pero mirando el punto que se encuentra a una cierta distancia X e Y del punto donde se encuentra el cursor gráfico.

Las funciones XPOS e YPOS

Estas dos funciones nos indican, en cualquier momento, la posición del cursor gráfico en la pantalla en coordenadas reales. Esto significa que si hemos definido un nuevo origen, con ORIGIN, si el cursor gráfico está en dicho origen, al preguntar por XPOS e YPOS, el ordenador nos dará sus coordenadas reales, y no las nuevas.

Algunas deficiencias en el BASIC del AMSTRAD

A la hora de ponernos a dibujar con nuestro AMSTRAD, echamos en falta funciones como CIRCLE (para dibujar arcos, círculos y elipses) y PAINT (para rellenar figuras cerradas). No nos queda más remedio que realizar nosotros mismos estas funciones. En el caso del CIRCLE es muy sencillo y se puede hacer una rutina que nos dibuje cualquier tipo de círculo, arco o elipse. En el caso de la sentencia PAINT es más complicado, porque realizar un programa en BASIC que me rellene cualquier figura cerrada, por compleja que sea, de un cierto color, sería muy lento. Por ello, se propone al lector que él mismo cree sus propias rutinas de PAINT según sus necesidades en cada momento.

A continuación aparece un programa que podemos utilizar para dibujar círculos. Lo único que hay que mandarle al programa es el centro, el radio y un número entre 0 y 2 que significa lo siguiente:

- 0 Dibujar un círculo.
- 1 Dibujar una elipse.
- 2 Dibujar un arco de circunferencia.

En el caso de que mandemos en uno (1), también tendremos que decir qué relación hay entre el radio horizontal y el vertical.

MANEJO DE SPRITES Y ELEMENTOS GRAFICOS

Si lo que mandamos es un dos (2), entonces, aparte de mandar la relación entre los radios, habrá que decir en qué ángulo (en grados) empezamos a dibujar y en qué ángulo (también en grados) terminamos. Las variables utilizadas son las siguientes:

X Coordenada X del centro.
Y Coordenada Y del centro.
R Radio del círculo.
S Relación entre radios.
A1 Angulo del que partimos.
A2 Angulo al que llegamos.

```

9000 REM *****
9010 REM *
9020 REM *   RUTINA PARA DIBUJAR CIRCULOS, ARCOS Y ELIPSES EN EL AMSTRAD *
9030 REM *   ----- *
9040 REM *
9050 REM * VARIABLES QUE HAY QUE PASARLE AL PROGRAMA *
9060 REM * ===== *
9070 REM *
9080 REM * X = COORDENADA X DEL CENTRO DE LA CIRCUNFERENCIA, ARCO O ELIPSE *
9090 REM * Y = COORDENADA Y DEL CENTRO DE LA CIRCUNFERENCIA, ARCO O ELIPSE *
9100 REM * R = RADIO DE LA CIRCUNFERENCIA, ARCO O ELIPSE *
9110 REM * S = EXCENTRICIDAD DE LA ELIPSE *
9120 REM * SW = 0,1,2 SEGUN LA SIGUIENTE TABLA *
9130 REM *       0 = CIRCULO *
9140 REM *       1 = ELIPSE *
9150 REM *       2 = ARCO *
9160 REM * A1 = ANGULO DESDE EL QUE EMPEZAMOS A DIBUJAR EL ARCO *
9170 REM * A2 = ANGULO HASTA EL QUE DIBUJAMOS EL ARCO *
9180 REM * *
9190 REM *****
9200 REM
9210 IF SW=0 THEN GOSUB 9250
9220 IF SW=1 THEN GOSUB 9370
9230 IF SW=2 THEN GOSUB 9640
9240 RETURN
9250 REM
9260 REM *****
9270 REM * DIBUJO DE CIRCULOS *
9280 REM *****
9290 REM
9300 FOR I=0 TO 6.5 STEP .2
9310   LET XO=X+R*COS(I)
9320   LET YO=Y+R*SIN(I)
9330   IF I=0 THEN PLOT XO,YO
9340   DRAW XO,YO
9350 NEXT I
9360 RETURN
9370 REM
9380 REM *****
9390 REM * DIBUJO DE ELIPSES *
9400 REM *****
9410 REM
9420 LET Z1=R*S
9430 LET Z2=R/S
9440 IF Z2>Z1 THEN LET SS=3*ABS(Z1/Z2):GOTO 9460
9450 LET SS=ABS(Z2/Z1)
9460 FOR XO=0 TO Z1 STEP SS
9470   LET YO=SQR(Z2^2*(Z1^2-XO^2)/Z1^2)
9480   IF XO=0 THEN PLOT X+XO,Y+YO
9490   IF XO=0 THEN PLOT X-XO,Y-YO
9500   IF XO=0 THEN PLOT X+XO,Y-YO
9510   IF XO=0 THEN PLOT X-XO,Y+YO:GOTO 9560
9520   PLOT X1,Y1:DRAW X+XO,Y+YO
9530   PLOT X2,Y2:DRAW X-XO,Y-YO

```

```

9540 PLOT X1,Y2: DRAW X+X0,Y-Y0
9550 PLOT X2,Y1: DRAW X-X0,Y+Y0
9560 LET X1=X+X0
9570 LET X2=X-X0
9580 LET Y1=Y+Y0
9590 LET Y2=Y-Y0
9600 NEXT X0
9610 PLOT X1,Y1: DRAW X1,Y2
9620 PLOT X2,Y1: DRAW X2,Y2
9630 RETURN
9640 REM
9650 REM *****
9660 REM * DIBUJO DE ARCOS *
9670 REM *****
9680 REM
9690 LET A=A1
9700 LET B=A2
9710 LET SS=.1
9720 IF A2<A1 THEN LET SS=-SS
9730 FOR I=A TO B STEP SS
9740 LET X0=X+R*COS(I)
9750 LET Y0=Y+R*SIN(I)
9760 IF I=A THEN PLOT X0,Y0
9770 DRAW X0,Y0
9780 NEXT I
9790 RETURN

```

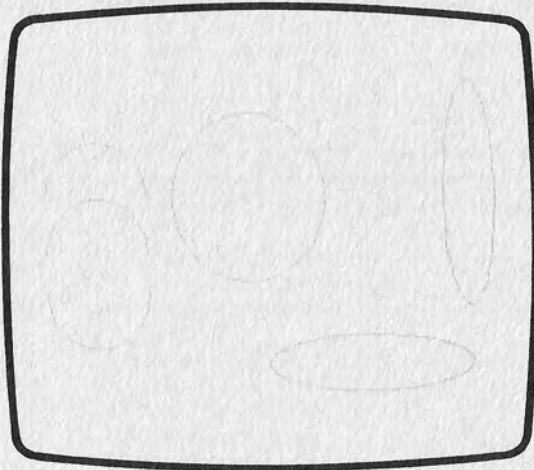


Fig. 7.

TRUCOS Y RUTINAS BASICAS



NA vez que hemos visto la alta resolución en el IBM pc y compatibles, estamos en disposición de entender el programa que se da a continuación. Con él el usuario podrá dibujar sus propias pantallas, o trozos de pantalla, para usarlas como presentación a sus programas.

Este programa nos permitirá dibujar puntos, rectas, rectángulos, etc., con unos po-

cos comandos. Por otro lado, al estar hecho en BASIC, permite que el propio lector modifique el programa a su gusto y según sus necesidades.

Al principio del programa el ordenador nos preguntará qué tipo de resolución horizontal queremos. Nos propone dos opciones, una de 320 puntos por línea y otra de 640. En el primero de los tipos el ordenador nos permite disponer de tres colores distintos para la tinta. En el segundo tipo de pantalla, a cambio de tener más resolución, sólo tenemos un color de tinta.

```

10 REM *****
11 REM *
12 REM * SSS U U PPPP EEEEE RRRR GGG RRRR AAA FFFFF III CCC SSS *
13 REM * S S U U P P E E R R G G R R A A F F I C C S S *
14 REM * S S U U P P E E R R G G R R A A F F I C C S S *
15 REM * SSS U U PPPP EEE RRRR G GG RRRR AAAAA FFF I C SSS *
16 REM * S S U U P P E E R R G G R R A A F F I C S S *
17 REM * S S U U P P E E R R G G R R A A F F I C C S S *
18 REM * SSS UUU P EEEEE R R GGG R R A A F III CCC SSS *
19 REM *
20 REM *****
21 REM
22 REM *****
23 REM ***** REALIZADO POR FRANCISCO MORALES GUERRERO *****
24 REM *****
25 REM
26 REM *****
27 REM ***** (c) EDICIONES SIGLO CULTURAL, 1987 *****
28 REM *****
29 REM
30 REM * INICIALIZACION DEL PROGRAMA *
31 REM
32 KEY OFF
33 FOR RT=1 TO 10
34 KEY(RT) OFF
35 NEXT RT
36 SCREEN 1
37 COLOR 16,1,,3
38 WIDTH 80
39 LET X=100
40 LET Y=100
41 LET X1=0
42 LET Y1=0
43 LET SS=1
44 DIM A%(10000),B%(10000),C%(3000),D%(2000),E%(2000)
45 GOSUB 160

```

TRUCOS Y RUTINAS BASICAS

```

46 CLS
47 LOCATE 21,1
48 INPUT "Numero de puntos en X (320 o 640) = ";A
49 IF A > 320 AND A < 640 THEN 47 ELSE IF A=320 THEN WIDTH 40:LET SW=1 ELSE LET SW
=2
50 REM
51 REM *****
52 REM * PROGRAMA PRINCIPAL *
53 REM *****
54 REM
55 LET PP=0
56 LET II=1
57 COLOR PP,II
58 CLS
59 GOSUB 92
60 GET (0,0)-(319*SW,199),B%
61 P=POINT(X,Y):IF P<>0 THEN P1=0 ELSE P1=1
62 PSET(X,Y),P1
63 A$=INKEY$:IF A$="" THEN 63
64 A=ASC(A$)
65 IF LEN(A$)=1 THEN 70
66 IF MID$(A$,2,1)="K" THEN PSET(X,Y),P:X=X-SS-SS*(X-SS<0):GOTO 61
67 IF MID$(A$,2,1)="M" THEN PSET(X,Y),P:X=X+SS+SS*(X+SS>319*SW):GOTO 61
68 IF MID$(A$,2,1)="H" THEN PSET(X,Y),P:Y=Y-SS-SS*(Y-SS<0):GOTO 61
69 IF MID$(A$,2,1)="P" THEN PSET(X,Y),P:Y=Y+SS+SS*(Y+SS>191):GOTO 61
70 IF A=83 OR A=115 THEN M$="ESCALA ACTUAL =" +STR$(SS)+"
I
INTRODUZCA NUEVA ESCALA (1-9)":LO=2:MA$="9":MI$="1":GOSUB 116:IF VAL(D$)>9 AND D$
<>" THEN 70 ELSE SS=VAL(D$)
71 IF A=80 OR A=112 THEN PSET(X,Y),II:X1=X:Y1=Y:SWW=1:GOTO 61
72 IF A=79 OR A=111 THEN PSET(X,Y),PP:X1=X:Y1=Y:SWW=2:GOTO 61
73 IF A=67 OR A=99 THEN LO=2:MA$="4":MI$="0":M$="COLOR":GOSUB 116:IF VAL(D$)>4 A
ND D$<>" THEN 72 ELSE II=VAL(D$)
74 IF A=76 OR A=108 THEN PSET(X,Y),P:GOSUB 92:LINE(X1,Y1)-(X,Y),-II*(SWW=1)-PP*(
SWW=2):X1=X:Y1=Y:GOTO 61
75 IF A=70 OR A=102 THEN PSET(X,Y),P:GOSUB 92:DD=POINT(X,Y):FOR I=X TO 319*SW:IF
POINT(I,Y)<>DD THEN DD=POINT(I,Y):PAINT(X,Y),II,DD:GOTO 61 ELSE NEXT I:GOTO 61
76 IF A=27 THEN PSET(X,Y),P:GOSUB 98:GOTO 61
77 IF A=75 OR A=107 THEN PSET(X,Y),P:GET (0,0)-(319,191),B%:GOTO 61
78 IF A=74 OR A=106 THEN PSET(X,Y),P:PUT (0,0),B%,PSET:GOTO 61
79 IF A=81 OR A=113 THEN PSET(X,Y),P:M$="( TERMINAMOS EL PROGRAMA ? (S/n)":MA$="
e":MI$="A":LO=2:GOSUB 116:IF D$<>"n" AND D$<>"N" AND D$<>"S" THEN 79 ELSE IF D$=
"S" THEN 139
80 IF A=90 OR A=122 THEN PSET(X,Y),P:M$="( BORRAMOS LA PANTALLA ? (S/n)":MI$="A"
:MA$="e":LO=2:GOSUB 116:IF D$<>"N" AND D$<>"n" AND D$<>"S" THEN 80 ELSE IF D$="S
" THEN CLS:GOTO 61
81 IF A=68 OR A=100 THEN PSET(X,Y),P:GOSUB 92:N$="DIRECTORIO":GOSUB 153:GOSUB 14
5:PRINT "PULSA UNA TECLA":A$=INPUT$(1):GOSUB 98:GOTO 61
82 IF A=66 OR A=98 THEN PSET(X,Y),P:GOSUB 92:N$="BORRAR UN PROGRAMA":GOSUB 153:G
OSUB 145:INPUT "NOMBRE DEL FICHERO = ",A$:KILL A$:GOSUB 98:GOTO 61
83 IF A=82 OR A=114 THEN PSET(X,Y),P:GOSUB 92:N$="RENOMBRAR UN PROGRAMA":GOSUB 1
53:GOSUB 145:INPUT "NOMBRE ANTIGUO DEL FICHERO = ",A$:PRINT:INPUT "NOMBRE MODERNO
DEL FICHERO = ",B$:NAME A$ AS B$:GOSUB 98:GOTO 61
84 IF A=77 OR A=109 THEN PSET(X,Y),P:DEF SEG=&HB800:M$="NOMBRE PARA GRABAR = ":M
A$="e":MI$="A":LO=12:GOSUB 116:IF D$="" THEN GOTO 61 ELSE BSAVE D$,0,16300:GOTO
61
85 IF A=78 OR A=110 THEN PSET(X,Y),P:DEF SEG=&HB800:M$="NOMBRE PARA GARGAR = ":M
A$="e":MI$="A":LO=12:GOSUB 116:IF D$="" THEN GOTO 61 ELSE BLOAD D$:GOTO 61
86 IF A=88 OR A=120 THEN PSET(X,Y),P:GOSUB 104:LOCATE 22,1:PRINT "X=";X;" ";Y=
";Y:LOCATE 23,1:PRINT "PULSA UNA TECLA":A$=INPUT$(1):GOSUB 110
87 IF A=63 THEN PSET(X,Y),P:GOSUB 160
88 IF (A=87 OR A=119) AND SSW=0 THEN PSET(X,Y),P:X2=X:Y2=Y:SSW=1:GOTO 61
89 IF (A=87 OR A=119) AND SSW=1 THEN PSET(X,Y),P:GET (X2,Y2)-(X,Y),D$:AX=X-X2:AY
=Y-Y2:X=X2:Y=Y2:GET (X,Y)-(X+AX,Y+AY),E$:SSW=0:GOTO 191
90 PSET(X,Y),P:GOTO 61
91 REM

```

```

92 REM *****
93 REM * ALMACENA PANTALLA *
94 REM *****
95 REM
96 GET (0,0)-(319*SW,199),A%:RETURN
97 REM
98 REM *****
99 REM * RECUPERA PANTALLA *
100 REM *****
101 REM
102 PUT (0,0),A%,PSET:RETURN
103 REM
104 REM *****
105 REM * ALMACENA 3 LINEAS INFERIORES *
106 REM *****
107 REM
108 GET (0,160)-(319*SW,199),C%:RETURN
109 REM
110 REM *****
111 REM * RECUPERA 3 LINEAS INFERIORES *
112 REM *****
113 REM
114 PUT (0,160),C%,PSET:RETURN
115 REM
116 REM *****
117 REM * ENTRADA DE DATOS *
118 REM *****
119 REM
120 GOSUB 104
121 LOCATE 21,1
122 PRINT SPC(120*SW);
123 LET D$=""
124 LET LO=0
125 LOCATE 22,1
126 PRINT M$;" " : CHR$(29);
127 C$=INKEY$
128 IF C$="" THEN GOTO 127
129 IF C$=CHR$(8) AND LO>0 THEN LO=LO-1:D$=LEFT$(D$,LO):PRINT CHR$(29);" _ ";CHR$(29);CHR$(29);
130 IF C$=CHR$(13) THEN GOTO 137
131 IF C$=" " OR C$="." THEN GOTO 133
132 IF C$>MA$ OR C$<MI$ THEN GOTO 127
133 PRINT C$;" _ ";CHR$(29);
134 LET D$=D$+C$
135 LET LO=LO+1
136 IF LO<>LO THEN GOTO 127
137 PRINT " _ ":GOSUB 110:RETURN
138 REM
139 REM *****
140 REM * SALIDA DEL PROGRAMA *
141 REM *****
142 REM
143 GOSUB 104:LOCATE 22,1:PRINT "PULSA [ESCAPE] PARA SALIR":PRINT "OTRA TECLA CO
NTINUAR":A$=INPUT$(1):IF ASC(A$)=27 THEN CLS:SYSTEM ELSE GOSUB 110:GOTO 61
144 REM
145 REM *****
146 REM * DIRECTORIO *
147 REM *****
148 REM
149 CLS:LOCATE 1,1:PRINT N$;" DE LA UNIDAD ";D$;" : "
150 FILES D$+"*.*":PRINT:PRINT
151 RETURN
152 REM
153 REM *****
154 REM * PEDIR UNIDAD DE DISCOS *
155 REM *****

```

TRUCOS Y RUTINAS BASICAS

```

156 REM
157 PSET(X,Y),P:M$=N$+" DE LA UNIDAD = ":M$="z":M$="A":L0=2:GOSUB 116:IF LEN(D
$)-2 THEN 157 ELSE IF D$="" THEN D$="A"
158 RETURN
159 REM
160 REM *****
161 REM * INSTRUCCIONES *
162 REM *****
163 REM
164 GOSUB 92:CLS:LOCATE 1,1:PRINT " COMANDOS DEL PROGRAMA SUPERGRAFICOS. "
165 PRINT:PRINT "<Q> ..... Finaliza sesion"
166 PRINT "<P> ..... Dibuja un punto (PSET)"
167 PRINT "<O> ..... Borra un punto (PRESET)"
168 PRINT "<L> ..... Traza una linea (LINE)"
169 PRINT "<X> ..... Visualiza coordenadas"
170 PRINT "<M> ..... Graba pantalla (SAVE)"
171 PRINT "<N> ..... Carga pantalla (LOAD)"
172 PRINT "<K> ..... Graba pantalla en memoria"
173 PRINT "<J> ..... Carga pantalla de memoria"
174 PRINT "<ECC> ..... Anula la ultima accion (DEL)"
175 PRINT "<Z> ..... Borra la pantalla (CLS)"
176 PRINT "<B> ..... Borra un fichero (KILL)"
177 PRINT "<D> ..... Muestra el directorio (FILES)"
178 PRINT "<R> ..... Renombra un fichero (RENAME)"
179 PRINT "<C> ..... Varia la velocidad del cursor"
180 PRINT "<CO> ..... Cambia color de tinta (COLOR)"
181 PRINT "<F> ..... Colorea una zona (FILL)"
182 PRINT "<W> ..... Copia parte del dibujo (WIND)"
183 PRINT "<?> ..... Muestra esta pantalla"
184 PRINT "<2,4,6,8> Mueven el cursor"
185 PRINT:PRINT "PULSA UNA TECLA":A$=INPUT$(1):GOTO 98
186 REM
187 REM *****
188 REM * CREACION DE UNA VENTANA *
189 REM *****
190 REM
191 A$=INKEY$:IF A$="" THEN 191
192 IF A$=CHR$(13) THEN GOTO 61
193 IF MID$(A$,2,1)="K" THEN PUT (X,Y),E$,PSET:X=X-SS-SS*(X-SS<0):GOTO 198
194 IF MID$(A$,2,1)="M" THEN PUT (X,Y),E$,PSET:X=X+SS+SS*(X+SS>319*SW-AX):GOTO 1
98
195 IF MID$(A$,2,1)="H" THEN PUT (X,Y),E$,PSET:Y=Y-SS-SS*(Y-SS<0):GOTO 198
196 IF MID$(A$,2,1)="P" THEN PUT (X,Y),E$,PSET:Y=Y+SS+SS*(Y+SS>199-AY):GOTO 198
197 GOTO 191
198 GET (X,Y)-(X+AX,Y+AY),F$:PUT (X,Y),D$,FSET:GOTO 191

```



Fig. 1. Resumen de comandos.

Veamos ahora qué comandos nos facilita el programa y cómo utilizarlos:

Q. FINALIZA SESION. Salimos del programa del BASIC y nos volvemos al sistema

operativo. El ordenador nos pide confirmación antes de hacerlo.

P. DIBUJA UN PUNTO. Ilumina el píxel sobre el que se encuentra el cursor gráfico. Si dicho punto ya está iluminado, no hay ningún cambio. El color de dicho punto coincidirá con el del color que se esté usando en ese momento.

O. BORRA UN PUNTO. Nos borra el punto sobre el que está situado el cursor gráfico. Lo pone del color del papel.

L. TRAZA UNA LINEA. Nos dibuja una línea desde el último punto donde se dibujó un punto (o se trazó una línea) hasta la posición actual del cursor gráfico. Si se usó la P para trazar el primer punto, la línea dibujará.

Si se utilizó la O, entonces la línea borrará a su paso.

X. **VISUALIZA COORDENADAS.** Nos permite ver en la parte inferior de la pantalla las coordenadas actuales (X e Y) del cursor gráfico.

M. **GRABA PANTALLA.** Para grabar la pantalla en disco. Nos preguntará el nombre que queremos darle. Hay que incluir la extensión si no queremos que el ordenador le ponga la extensión.

N. **CARGA PANTALLA.** Nos permite leer y cargar una pantalla almacenada en una variable de forma que, si seguimos trabajando en la pantalla y queremos volver a ver cómo teníamos las cosas antes de hacer ciertas modificaciones, podemos recuperarla.

J. **CARGA PANTALLA DE LA MEMORIA.** Nos muestra la pantalla que habíamos almacenado en memoria mediante la opción K.

ESC. **ANULA LA ULTIMA ACCION.** Pulsando la tecla ESCAPE anulamos la última acción que hayamos realizado y volvemos a ver la pantalla como se encontraba hace unos instantes.

Z. **BORRA LA PANTALLA.** Borra todo el contenido gráfico de la pantalla. Esta opción pide confirmación antes de ejecutarse.

B. **BORRA UN FICHERO.** Borra cualquier fichero que se encuentre en el disco. Nos imprime el directorio antes de pedirnos el nombre. Hay que incluir la extensión del fichero a borrar.

D. **MUESTRA DIRECTORIO.** Nos pide la unidad de la cual queremos ver el directorio.

R. **RENOMBRA UN FICHERO.** Es una opción muy útil para renombrar aquellos archivos de pantallas que, por error, al no darle extensión, tienen extensión .BAS. Nos muestra el directorio de la unidad de disco que nosotros le digamos.

S. **VARIA LA VELOCIDAD DEL CURSOR.** Para hacer que el cursor gráfico se mueva más deprisa por la pantalla. El valor de la velocidad puede variar entre uno (1) y nueve (9).

C. **CAMBIA EL COLOR DE LA TINTA.** A partir del momento en que cambiemos el color de la tinta, todo lo que dibujemos aparecerá con el nuevo color. Esta opción no causa ningún efecto cuando estamos en el modo de 640 puntos por línea.

F. **COLOREA UNA ZONA.** Nos permite llenar de color una zona que esté cerrada, como puede ser un rectángulo. Para hacerlo,

hay que colocar el cursor gráfico dentro de la zona cerrada a colorear y pulsar la tecla F. Los límites de la zona a colorear sólo pueden ser de un color.

W. **COPIA PARTE DEL DIBUJO.** Nos permite copiar una parte de la pantalla en otra parte de la misma. Para ello tenemos que colocar el cursor gráfico sobre la esquina superior izquierda de la parte a copiar y pulsar la W. A continuación nos trasladamos a la parte inferior derecha de la zona a copiar y volvemos a pulsar la tecla W. A partir de este momento, y usando las teclas del cursor, podemos mover esta zona por la pantalla y colocarla donde deseemos. Para fijarla en una parte de la pantalla no hace falta nada más que pulsar la tecla ENTER.

? Muestra una pantalla con un resumen de comandos.



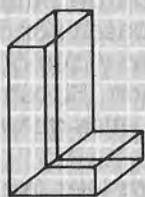
Fig. 2. Ejemplo de pantalla.

Para mover el cursor gráfico por la pantalla se utilizan las teclas del cursor que están en el teclado numérico. Como dicho teclado tiene dos funciones distintas, tiene que estar de forma que al pulsar las teclas del cursor fuera del programa el cursor se mueva. La luz de NUM LOCK ha de estar apagada.



Fig. 3. Uso de la función WINDOW (W).

EL TALLER DEL HARDWARE



A entrada de datos a través de la pantalla es, sin duda, una de las más atractivas para todo ordenador personal. Si observamos cómo se produce físicamente la representación de caracteres y gráficos en la pantalla, veremos que se efectúa mediante la activación de puntos de mayor o menor intensidad, sobre un barrido uniforme. Puede verse en la figura cómo se lleva a cabo el barrido del rayo de electrones sobre la pantalla que, al estar compuesta de sales de fósforo, transforma la intensidad de los electrones en emisión secundaria luminosa. La emisión luminosa existe incluso en las zonas de la pantalla que aparecen oscuras. En las pantallas de color, siempre que exista un fondo distinto del negro, la emisión resulta fácilmente detectable mediante alguno de los dispositivos sensibles a la luz, como las resistencias LDR y los fotodiodos.

LINEA VISIBLE



RETORNO NO VISIBLE

Fig. 1. Barrido de la pantalla.

Conviene hacer una distinción inicial entre los dos procedimientos básicos para detectar un punto luminoso en la pantalla. El método rápido detecta la emisión luminosa del primer barrido que se produzca delante del sensor después de haber activado un pulsador, y el método lento, mediante el barrido programado de un carácter por la pantalla, parado en cada posición durante un tiempo suficiente, hasta detectar que el lápiz se encuentra encima de un punto determinado.

Para hacer el método rápido se deben usar dispositivos sensibles a la luz de características muy rápidas y muy sensibles. Para comprender la dificultad del problema veamos los valores necesarios para su funcionamiento. La duración del barrido de una línea en la pantalla, tanto en sistema PAL como en NTSC, que son los que normalmente se utilizan en las pantallas de color de los ordenadores personales, es de 64 microsegundos. De este tiempo hay que descontar la duración del pulso de sincronismo y unos espacios denominados "porches", que resultan necesarios para el centrado de la imagen, se han de barrer en unos 50 microsegundos. El número de puntos visibles físicamente en la pantalla en una línea es de 256 a 640, por tanto, el tiempo para cada punto es de unos 80 a 200 nanosegundos. Si pretendemos detectar la presencia de un punto iluminado de esta duración es necesario que el elemento sensible alcance un valor apreciable y tenga la rapidez suficiente para poder indicar el momento en que se produce el paso.

Algunos ordenadores personales, como el IBM-PC, presentan una entrada para lápiz óptico, en el adaptador de color y gráficos, que llega hasta una de las patillas del controlador de vídeo MCM6845. Este circuito integrado del controlador de vídeo posee internamente la circuitería necesaria para realizar la lectura de la entrada por lápiz óptico. Esta cir-

EL TALLER DEL HARDWARE

cuitería consiste en un contador que se encuentra parado y que se conecta a la entrada del reloj de activación de barrido al cerrarse el pulsador de lectura y producirse el comienzo de barrido del cuadro. El contador se detiene si se produce pulso de entrada del lápiz óptico. El contador realiza internamente el barrido de todas las columnas de una fila y sucesivamente todas las filas, desde arriba hasta abajo. Al terminar una exploración completa sin activación, vuelve a empezar la cuenta.

Se puede leer el contador para conocer en qué fila-columna se produjo la activación del lápiz. El programa deberá estar consultando el bit de estado para saber cuándo se detiene el contador. Generalmente el lenguaje propio de la máquina posee instrucciones para realizarlo automáticamente.

La circuitería necesaria para el lápiz se explica a continuación. Es necesario un diodo PIN muy rápido, a ser posible con lente concentradora. El amplificador debe tener un tiempo de subida muy pequeño, para poder captar la señal luminosa de la pantalla. El pulsador indicador de lectura puede colocarse sobre el cuerpo del lápiz para comodidad o bien integrarse en la punta para que se active al presionar sobre la pantalla.

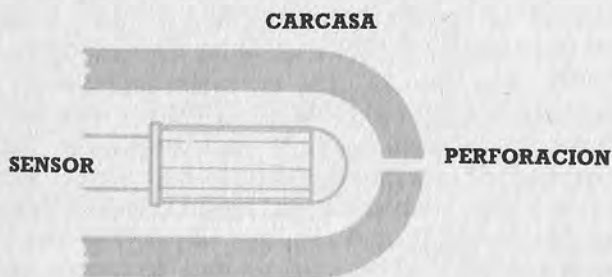


Fig. 2. Punta del lápiz óptico.



Fig. 3. Esquema práctico de montaje con lente concentradora.

Observando la señal de salida del sensor óptico, veremos una secuencia de pulsos, correspondiente a las líneas que pasan por delante de la lente del lápiz luminoso y que quedan dentro de su campo de visión. El contador se parará ante la primera de ellas que sobrepase el umbral fijado.

El programa necesario para leer la posición del punto donde se encuentra el lápiz y que puede utilizarse como test es, por ejemplo, para el caso del IBM-PC:

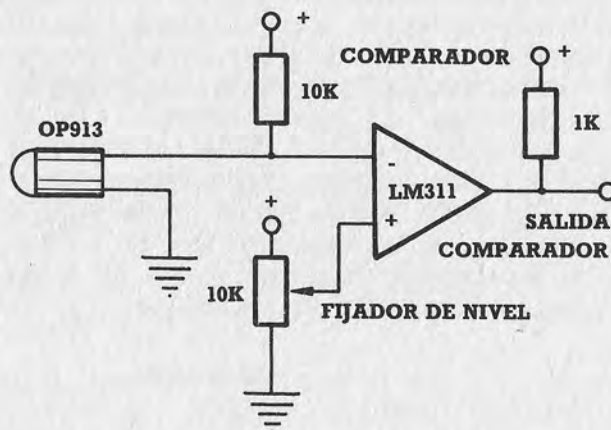


Fig. 4. Circuito de lectura.

```

10 REM TESTPEN
20 REM PRUEBA DE LAPIZ OPTICO, PARA CALIBRACION
30 REM PARA PEN RAPIDO Y IBM-PC
40 XA=0 : YA=0
50 KEY OFF
60 CLS
70 ON PEN GOSUB 180: REM INDICACION DE ENTRADA POR LAPIZ
80 SCREEN 1: REM GRAFICOS DE MEDIA RESOLUCION
90 COLOR 1,1
100 PRINT "PRUEBA DE LAPIZ OPTICO"
110 PRINT "PULSE SOBRE LA PANTALLA"
120 PEN ON: REM ACTIVA ENTRADA POR LAPIZ
130 FOR I=1 TO 10000: REM BUCLE
140   A=A+1
150 NEXT I
160 CLS : PRINT "TEST TERMINADO"
170 END
175 REM  RUTINA DE LECTURA CADA VEZ QUE SE PULSE EL LAPIZ

```

```

180 X=PEN(1): REM LEE LAS COORDENADAS
190 Y=PEN(2)
200 LINE (X,Y)-(XA,YA),2: REM PINTA LINEA
210 XA=X : YA=Y
220 LOCATE 24,1:PRINT X,Y;
230 RETURN

```

También se incluye un programa sencillo de demostración de las posibilidades del

lápiz óptico para la creación y edición de gráficos en la pantalla.

```

10 REM JUEGOPEN
20 REM DEMOSTRACION DEL EMPLEO DE LAPIZ OPTICO RAPIDO
25 REM PARA IBM-PC
30 DEFINI A:DIM A(200,2),CRUZ(100)
40 SC=1:LIMITEY=319:LIMITEC=290:LIMITEY=190
45 COLR=0:COLRF=1
50 SCREEN SC,0:J=2:A(1,1)=10:A(1,2)=10:A(1,1)=10
55 A(2,2)=10:CLS
58 REM ACTIVA TECLAS DE FUNCION Y CURSORES
60 KEY OFF: FOR I=1 TO 14: KEY(I) ON: NEXT I
70 ON KEY(1) GOSUB 360: ON KEY(2) GOSUB 400
80 ON KEY(5) GOSUB 460: ON KEY(6) GOSUB 490
85 ON KEY(7) GOSUB 720
90 ON KEY(8) GOSUB 640: ON KEY(10) GOSUB 670
95 ON KEY(14) GOSUB 600
100 ON KEY(11) GOSUB 420: ON KEY(12) GOSUB 520
105 ON KEY(13) GOSUB 560
110 LINE (1,5)-(10,5),1: LINE (5,2)-(5,9),1
120 GET (1,1)-(9,9),CRUZ
130 ON PEN GOSUB 200
140 GOSUB 300: REM PONE PANTALLA INICIAL
150 PEN ON: REM ACTIVA LAPIZ
160 S$=INKEY$: IF S$="" THEN 160
170 IF S$="F" OR S$="f" THEN WIDTH 80:END: " TERMINACION
180 IF S$="I" THEN GOSUB 750: " INVIERTE IMAGEN
190 GOTO 160
200 REM ENTRADA DE POSICION POR LAPIZ OPTICO
210 FOR I=1 TO 2:A(J,I)=PEN(I):NEXT I
215 IF A(J,2)>190 THEN A(J,2)=190
220 IF A(J,2)<4 THEN A(J,2)=4
230 IF A(J,1)<4 THEN A(J,1)=4
240 IF A(J,1)>LIMITEY-4 THEN A(J,1)=LIMITEY-4
250 PUT (A(J,1)-4,A(J,2)-4),CRUZ,XOR
260 PUT (A(J-1,1)-4,A(J-1,2)-4),CRUZ,XOR
270 J=J+1
280 RETURN
290 REM PONE LA PANTALLA EN BLANCO
300 CLS
310 LINE (0,0)-(LIMITEY,LIMITEY),COLRF,BF
320 LOCATE 25,1
330 PRINT "1-BURRA,2-LINEA,5-CIRC,7-RELL,8-RECT";
340 RETURN
350 REM BURRA LINEA ANTERIOR Y CAMBIA EL APUNTADOR
360 LINE (A(J-1,1),A(J-1,2))-(A(J-2,1),A(J-2,2)),COLRF
370 IF J<2 THEN RETURN ELSE J=J-1
380 RETURN
390 REM DIBUJA RECTA ENTRE LOS ULTIMOS PUNTOS DEFINIDOS
400 LINE (A(J-1,1),A(J-1,2))-(A(J-2,1),A(J-2,2)),COLR
410 RETURN
420 REM AJUSTE HACIA ARRIBA
430 PUT (A(J-1,1)-4,A(J-1,2)-4),CRUZ,XOR
435 A(J-1,2)=A(J-1,2)-1
440 PUT (A(J-1,1)-4,A(J-1,2)-4),CRUZ,XOR
450 RETURN
460 R=SQR((A(J-2,1)-A(J-1,1))^2+(A(J-2,2)-A(J-1,2))^2)
470 CIRCLE (A(J-1,1),A(J-1,2)),R,0
480 RETURN
490 R=SQR((A(J-2,1)-A(J-1,1))^2+(A(J-2,2)-A(J-1,2))^2)
500 CIRCLE (A(J-1,1),A(J-1,2)),R,1
510 RETURN
520 REM HACIA IZQUIERDA
530 PUT (A(J-1,1)-4,A(J-1,2)-4),CRUZ,XOR
535 A(J-1,1)=A(J-1,1)-1
540 PUT (A(J-1,1)-4,A(J-1,2)-4),CRUZ,XOR

```

EL TALLER DEL HARDWARE

```

550 RETURN
560 REM HACIA DERECHA
570 PUT (A(J-1,1)-4,A(J-1,2)-4),CRUZ,XOR
575 A(J-1,1)=A(J-1,1)+1
580 PUT (A(J-1,1)-4,A(J-1,2)-4),CRUZ,XOR
590 RETURN
600 REM HACIA ABAJO
610 PUT (A(J-1,1)-4,A(J-1,2)-4),CRUZ,XOR
615 A(J-1,2)=A(J-1,2)+1
620 PUT (A(J-1,1)-4,A(J-1,2)-4),CRUZ,XOR
630 RETURN
640 REM RECTANGULO
650 LINE (A(J-1,1),A(J-1,2))-(A(J-2,1),A(J-2,2)),0,B
660 RETURN
670 REM TECLA 10, CAMBIA COLOR PRINCIPAL AL SIGUIENTE
680 COLR=COLR+1
690 RETURN
700 REM
710 REM RELLENA
720 PUT (A(J-1,1)-4,A(J-1,2)-4),CRUZ,XOR
730 PAINT (A(J-1,1),A(J-1,2)),0
740 RETURN
750 REM INVIERTE IMAGEN EN BASIC PARA SACAR EN IMPRESORA
760 DEF SEG=$H800
770 FOR I1=0 TO $H2000
780   POKE I1,255 XOR PEEK(I1)
790   POKE I1+$H2000,255 XOR PEEK(I1+$H2000)
800 NEXT I1: DEF SEG: RETURN
810 FOR I1=0 TO 359
820   FOR J1=0 TO 190
830     PSET (I1,J1),1 XOR POINT(I1,J1)
840   NEXT J1
845 NEXT I1
850 RETURN

```

El método lento requiere mucha menos circuitería, pues puede conectarse directamente a una de las entradas de la tarjeta de ampliación de puertos. No permite las mismas operaciones que el rápido, pues es necesario hacer por programa lo que el método rápido hace por circuitería.

En este método la exploración de fila y columna es preciso realizarla por programa, presentando durante un período suficiente una fila o columna luminosa sobre fondo oscuro. El ancho de la fila o columna podría hacerse de más de una línea, para mayor facilidad de lectura. El programa de lectura de la posición del lápiz deberá conservar en memoria el contenido que tiene la pantalla en ese momento, realizar el barrido y volver a restaurar la pantalla inicial, indicando la nueva posición del lápiz mediante algún símbolo de cursor.

■ Circuito de lápiz para el Spectrum

El circuito consta de un sensor óptico con un amplificador y de un pulsador con el cual indicamos cuándo queremos leer los datos. Los datos que podemos leer en el puerto de entrada son los siguientes:

- Pulsador abierto sensor no iluminado . 255
- Pulsador cerrado sensor no iluminado 254

- Pulsador abierto sensor iluminado 253
- Pulsador cerrado sensor iluminado 252

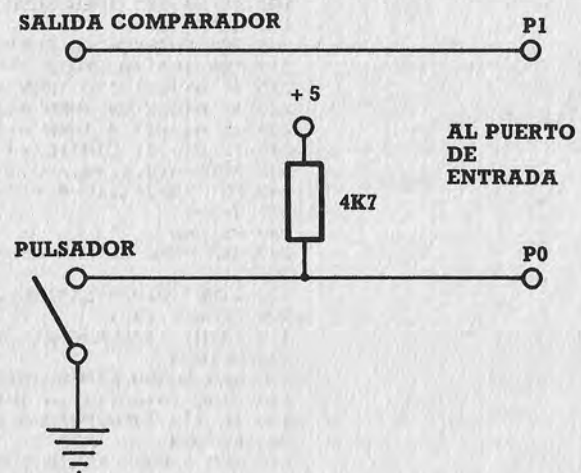


Fig. 5. Circuito de conexión al Spectrum.

Los ocho bits del bus de datos dan la señal "1" binaria mientras no pulsemos el botón del lápiz ni iluminemos el sensor, ya que están conectados a +5v. Si pulsamos el botón, el bit 0 se pondrá a "0" lógico. Si el sensor es iluminado con cierta intensidad (determinada por P1), el bit 1 se pondrá a "0" lógico. El amplificador debe tener un tiempo de subida muy pequeño para poder captar la señal lumi-

nosa de pantalla. Este circuito corresponde al de un lápiz óptico lento y su resolución vendría determinada por el modo de excitación del sensor: puede ser a través de una sencilla perforación (fig. 2) o de una lente concentradora (fig. 3), con la que se aumentaría la resolución. Para el empleo del lápiz óptico se re-

comienda utilizar un alto grado de contraste en el ajuste de la iluminación de la pantalla.

Programas para el Spectrum

A continuación se muestran unos programas desarrollados en el Spectrum:

```
10 REM PROGRAMA 1
20 BORDER 7: PAPER 0: INK 7: CLS
30 PRINT AT 10,10: IN X: REM LECTURA DEL PUERTO LAPIZ OPTICO
40 GO TO 30
```

Con el *programa 1* comprobamos si el circuito está bien realizado: introducimos y ejecutamos el programa. A continuación colocamos el lápiz óptico en algún punto oscuro de la pantalla y mantenemos el botón sin pulsar. El valor registrado en la pantalla debe ser 255. Ahora pulsamos el botón y debemos obtener el valor 254. Entonces colocamos el lápiz óptico en el borde blanco de la pantalla y sin el botón apretado el dato en la pantalla debe ser

253. A continuación apretamos el botón y el valor final tiene que ser 252. Si tras estas comprobaciones los resultados son positivos, el circuito está bien realizado. Si no, debemos comprobar si las conexiones se han realizado bien, si algún contacto eléctrico falla, o el interior de un cable está cortado. Si nada de esto sucede, puede que algún componente esté defectuoso y haya que sustituirlo.

```
10 REM PROGRAMA 2
20 INK 7: PAPER 0: BORDER 7: BRIGHT 1
30 CLS: FOR X=0 TO 21
40 PRINT AT Y,0: "=====":REM 32 CUADRADOS
50 IF IN X=252 THEN GO TO 70: REM LECTURA DEL L.O. PARA RASTREO Y
60 NEXT Y: GO TO 30
70 CLS: FOR X=0 TO 31
80 PRINT AT Y,X: " ": REM UN CUADRADO
90 IF IN X=252 THEN GO TO 110: REM LECTURA DEL L.O. PARA RASTREO X
100 NEXT X: GO TO 70
110 CLS: PRINT "COORDENADAS: X=":X: " Y=":Y
120 STOP
```

El *programa 2* es un ejemplo de aplicación directa para obtener las coordenadas a las que apunta el lápiz óptico a través del BASIC. Los colores se cambian en la línea 20 y ello se hace, pues, con esos nuevos atributos (tinta blanca con brillo, papel negro y borde blanco) el contraste es mucho mayor. En la línea 30 borramos la pantalla e iniciamos el bucle que rastrea la coordenada Y. Esto se hace con una barra de 32 cuadrados de larga a lo largo de las 22 filas del eje Y. Dentro del bucle tenemos la instrucción IN, que nos sirve para leer el puerto X y saber si el lápiz óptico está siendo iluminado. En tal caso (cuadro IN X = 252) se salta al segundo bucle de la línea 70. Aquí volvemos a borrar la pantalla para que lo que hubiera imprimido no nos interfiera en el rastreo por el eje X. En este bucle mo-

vemos un cuadrado de izquierda a derecha y, asimismo, con la instrucción IN comprobamos si el lápiz óptico es iluminado por el cuadrado. Cuando esto suceda ya tendremos las coordenadas X e Y en baja resolución.

Este programa se puede emplear como subrutina en un programa que emplee el lápiz óptico para que el usuario escoja entre los menús las opciones deseadas, ya que cada opción ocupa una línea de caracteres y basta con la baja resolución. Incluso podríamos suprimir el rastreo en el eje de la X si sólo hay una opción por cada línea.

He aquí un programa en código máquina que realiza la función de calcular las coordenadas donde está el lápiz óptico, pero a una velocidad mucho mayor que la del BASIC y en alta resolución:

```

DI
LD BC,6144
LD HL,16384
A LD (HL),255
IN A,(X)      X=DIRECCION DEL PUERTO DEL L.OPTICO
CP 252
JR NZ B
INC HL
DJ NZ A
EI
RET
B LD (51200),HL
LD (HL),1
LD BC,8
C SLA (HL)
IN A,(X)
OR 032D
DJ NZ C
EI
RET
D LD A,B
LD (51202),A
EI
RET

```

Las instrucciones DI y EI se emplean para que la ULA no nos interrumpa mientras ejecutamos el programa. El bucle etiquetado con "A" se encarga de realizar el primer rastreo. Este bucle consiste en ir "pokeando" con el valor 255 desde la dirección 16384 (donde empieza la memoria de pantalla) hasta la dirección 22528 (donde acaba la memoria de pantalla y empieza la memoria de atributos o colores). A la vez leemos el puerto X, que es donde tenemos conectado el lápiz óptico. Cuando éste nos dé la señal de que ha sido iluminado, detenemos el contador y "pokeamos" en la dirección 51200, 51201 la posición X, Y relativa (pues en base decimal no podemos usar ese valor directamente como coordenada) a la que apunta el lápiz óptico (esto es lo que hace la parte del programa etiquetada con la letra B). Entonces tomamos el byte de la pantalla al que apunta el lápiz óptico y "pokeamos" en él un 1 decimal, que en binario sería 00000001. Ese 1 significa un píxel o punto más pequeño de la pantalla. Ahora comienza el bucle etiquetado con la letra C, el cual hace lo siguiente: rota el byte (es decir, el antiguo dato binario 00000001 pasaría a ser 00000010),

comprueba si el lápiz óptico es iluminado, si no, continúa rotándolo; pero si es iluminado por nuestro píxel, entonces saltamos a la parte del programa etiquetada con D. Esta parte nos "pokea" en la dirección 51202 el número de veces que ha tenido que rotar el byte hasta conseguir la coordenada exacta del lápiz óptico.

La dirección del puerto de entrada no se ha fijado y se simboliza con "X". Finalmente, el programa "pokea" en las direcciones 51200, 51201 y 51202 unos valores a partir de los cuales podemos saber a dónde apuntaba el lápiz óptico. Las coordenadas serían las siguientes:

$$X = ((DIR - 16384)/24) + PEEK 51202$$

$$Y = 176 - ((DIR - 16384)/32)$$

En la expresión anterior DIR es la suma del valor encontrado en 51200 y del valor en 51201 multiplicado por 256.

Ahora mostramos un programa en BASIC para dibujar con el lápiz óptico que emplea la rutina en código máquina:

```

10 REM PROGRAMA DIBUJO
20 FOR A=60000 TO 60069
30 READ A
40 POKE A,B
50 NEXT A
60 POKE 23658,B: REM FIJA MAYUSCULAS
70 IF INKEY$="C" THEN GO SUB 140: REM COLORES
80 IF INKEY$="H" THEN GO SUB 220: REM CIRCULO
90 IF INKEY$="B" THEN OVER 1: REM BORRAR
100 IF INKEY$="P" THEN OVER 0: REM PINTAR

```

```

110 IF INKEY#="L" THEN GO SUB 420: REM LINEA
120 IF IN X=254 OR IN X=252 THEN GO SUB 340: REM PUNTO
130 GO TO 70
140 INPUT "TINTA?":L: INK L
150 INPUT "PAPEL?":C: PAPER C
160 INPUT "BORDE?":B: BORDER B
170 INPUT "BRILLO?":O: BRIGHT O
180 INPUT "PARPADEO?":F: FLASH F
190 INPUT "INVERTIDO?":I: INVERSE I
200 PRINT #0:"BORRO TODO?": PAUSE 0: IF INKEY#="S" THEN CLS
210 RETURN
250 PRINT #0:A: 0,0:"SELECCIONA RADIO"
260 IF IN X=254 OR IN X=252 THEN GO SUB 300: GO TO 280
270 GO TO 260
280 CIRCLE X1,Y1,R
290 RETURN
300 LET X1=X: LET Y1=Y
310 GO SUB 340
320 LET R=SQR ((ABS (X-X1)-2)+(ABS (Y-Y1)-2))
330 RETURN
340 RANDOMIZE USR 60058
350 OVER 0: INVERSE 0: FLASH 0: BRIGHT 1
360 INK 7: PAPER 0: BORDER 7: CLS
370 RANDOMIZE USR 60000
380 LET DIR=PEEK 51200+256*PEEK 51201
390 LET Y=176-((DIR-16384)/32)
400 LET X=(DIR-16384)/24+PEEK 51202
410 RANDOMIZE USR 60046: RETURN
420 PRINT #0:A: 0,0:"SELECCIONA PUNTO FINAL"
430 LET X1=X: LET Y1=Y
440 IF IN X=254 OR IN X=252 THEN GO SUB 340: GO TO 460
450 GO TO 440
460 INPUT "ARCO?":A: DRAW X-X1,Y-Y1,A
470 RETURN
480 DATA 243,1,0,24,33,0,64,54,255,219,X,254,252,32,6
490 DATA 35,16,245,251,201,34,0,200,54,1,1,8,0,203,38
500 DATA 219,X,254,252,32,5,16,246,251,201,120,50,2,200
510 DATA 251,201: REM DATOS C/M RUTINA LAPIZ OPTICO
520 DATA 17,0,64,33,0,128,1,0,27,237,176,201,17,0,128,33,0
530 DATA 64,1,0,27,237,176,201: REM C/M TRASBASE PANTALLA

```

Con este programa se pueden trazar puntos, líneas, arcos y círculos. Combinándolos podemos hacer pantallas de presentación y cambiarle los colores. La estructura del programa es sencilla: hay un bucle principal que interpreta si quieres modificar los colores (tecla C), trazar un círculo (tecla H), una línea (tecla L), borrar (tecla B), pintar (tecla P) o sim-

plemente pintar un píxel directamente con el lápiz. Todos los trazados emplean la subrutina que calcula las coordenadas a las que apunta el lápiz óptico.

Los colores se cambian pulsando la tecla C, entonces en la última línea de la pantalla imprimirá una serie de comandos:

```

TINTA?.....PODEMOS ESCOGER DE 0 A 7
PAPEL?....." " " " "
BORDE?....." " " " "
BRILLO?....." " ENTRE 0 Y 1
PARPADEO?....." " " " "
INVERTIDO?....." " " " "
BORRO TODO?....PULSAR LA TECLA S SI SE QUIERE BORRAR LA PANTALLA

```

Los colores son: 0-NEGRO, 1-AZUL, 2-ROJO, 3-VIOLETA, 4-VERDE, 5-CELESTE, 6-AMARILLO, 7-BLANCO.

Al final del programa hay un conjunto de datos que son la rutina de rastreo para el lápiz óptico y un intercambio de bloques de memoria para no perder la imagen de la pantalla cuando se buscan las coordenadas del lá-

piz óptico. El programa en ensamblador de las rutinas para el intercambio de memoria es el siguiente:

```

RUTINA QUE RETORNA LA PANTALLA
LD DE,16384
LD HL,32768

```

```
LD BC,6912
LDIR
RET

RUTINA QUE GUARDA LA PANTALLA
LD DE,32768
LD HL,16384
LD BC,6912
LDIR
RET
```

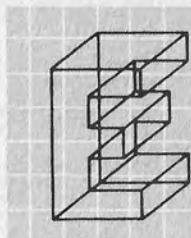
Estas rutinas se emplean para que el dibujo contenido en la pantalla no se pierda, pues la rutina que rastrea las coordenadas del lápiz óptico imprime en pantalla estropeando el dibujo que hubiera trazado en ella. Es decir, que la segunda rutina guarda la memoria donde está contenida la pantalla y la primera rutina devuelve de nuevo los datos a la pantalla. La dirección de la memoria de pantalla es la 16384 y la dirección donde guardamos la pantalla para que no se estropee es la 32768.

La longitud del bloque de memoria es 6912 bytes. En el programa BASIC la subrutina para leer el lápiz óptico está en la línea 340. Primero hace una RANDOMIZE USR 60058, que es una llamada a la rutina en código máquina situada en esa dirección. En esta dirección está la rutina que guarda la memoria de pantalla. Después cambia los colores: la tinta blanca con brillo y el papel negro para que exista contraste y el borde blanco para potenciar aún más el contraste. Entonces se procede a la llamada de la rutina que detecta la posición del lápiz óptico mediante RANDOMIZE USR 60000. Una vez que la rutina finaliza su tarea, retorna al BASIC y desde el BASIC hacemos unos cálculos finales para obtener las coordenadas definitivas. Es ahora cuando llamamos a la rutina de retorno de pantalla con RANDOMIZE USR 60046 para tener de nuevo nuestro dibujo en el televisor y continuar con nuestra labor de artistas.

APRENDER CON EL ORDENADOR

NATURALEZA Y TECNOLOGIA

Electrostática: Teoría de Gauss



El estudio de la electrostática se remonta a épocas relativamente recientes. Debemos su formulación a los experimentos de Priestley, en 1767, y repetidos por Coulomb, en 1785. Existen diversos aparatos básicos

donde se observa la existencia de cargas electrostáticas tales como el electroscopio, el electróforo o los generadores electrostáticos.

La carga eléctrica es una magnitud básica en la estructura del átomo. Una de las fuerzas más importantes en los átomos es la atracción entre las cargas opuestas del núcleo y de la corteza.

Campo eléctrico

Puede definirse el campo eléctrico en un punto dado del espacio como la fuerza por unidad de carga que actuaría sobre una carga dada situada en dicho punto.

Sabemos que la fuerza entre dos cargas cualesquiera es:

$$F \cdot \hat{a} = \frac{1}{4 \cdot \pi \cdot \epsilon} \cdot \frac{Q \cdot Q'}{R^2} \cdot \hat{a}$$

donde $\hat{a}$ es el vector director en la dirección de la línea que une las dos cargas.

Como el campo es por unidad de carga
=>

$$E \cdot \hat{a} = \frac{1}{4 \cdot \pi \cdot \epsilon} \cdot \frac{Q}{R^2} \cdot \hat{a}$$

El potencial eléctrico queda definido como:

$$V = \frac{1}{4 \cdot \pi \cdot \epsilon} \cdot \frac{Q}{R}$$

Aplicación del Teorema de Gauss a una esfera

Matemáticamente el teorema de Gauss se enuncia como:

$$\int E \cdot dS = \sum_i \frac{Q_i}{\epsilon_0}$$

La integración de $E \cdot dS$ a toda la superficie nos da el número de líneas de campo que atraviesan la superficie.

En el caso de la esfera, como el campo es constante para un mismo radio =>

$$E \cdot \int dS = \frac{Q}{\epsilon_0}$$

Luego, como el valor de la integral es la superficie de la esfera = $4 \cdot \pi \cdot R^2$ =>

$$E = \frac{Q}{4 \cdot \pi \cdot \epsilon_0 \cdot R^2}$$

Que es la fórmula utilizada en el programa, teniendo en cuenta siempre que el campo eléctrico en el interior de un conductor siempre es nulo.

```
10 REM*****
20 REM*      PROGRAMA      *
30 REM*      DE            *
40 REM*      ELECTROESTATICA *
50 REM*****
```

```

60 REM*   VALIDO PARA AMSTRAD  *
70 REM*   IBM-PC Y COMPATIBLES *
80 REM*   COMMODORE, SPECTRUM *
90 REM*   Y MSX                *
100 REM*****
110 PRINT "ESTE PROGRAMA CALCULA EL"
120 PRINT "POTENCIAL Y EL CAMPO "
130 PRINT "ELECTROESTATICO DE UNA ESFERA"
140 PRINT "DE RADIO VARIABLE EN UN PUNTO"
150 PRINT "CUALQUIERA DEL ESPACIO"
160 PRINT
170 PRINT
180 PRINT "INTRODUCE EL VALOR DE LA CARGA DE"
190 PRINT "LA ESFERA ":INPUT Q
200 PRINT "INTRODUCE EL RADIO DE LA ESFERA"
210 INPUT R
220 PRINT "INTRODUCE LA DISTANCIA DEL CENTRO DE"
230 PRINT "LA ESFERA AL PUNTO DE ESTUDIO"
240 INPUT D
250 PRINT "LOS DATOS SON:"
260 PRINT "RADIO=";R;" METROS"
270 PRINT "CARGA=";Q;" CULOMBIOS"
280 PRINT "DISTANCIA=";D;" METROS"
290 PRINT
300 PRINT
310 LET PI=3.141532
320 REM**DEFINO EL EPSILON PARA EL VACIO**
330 LET EP=1
340 IF D< R THEN GOTO 390
350 PRINT "EL CAMPO ELECTRICO ES :";Q/(4*PI*EP*D*D);" VOL/M."
360 PRINT
370 PRINT "EL POTENCIAL ES :";Q/(4*PI*EP*D);" VOL"
380 GOTO 430
390 REM*****SI EL PUNTO ESTA DENTRO DE LA ESFERA****
400 PRINT "EL CAMPO ELECTRICO ES :";0
410 PRINT
420 PRINT "EL POTENCIAL ELECTRICO ES :";Q/(4*PI*EP*R);" VOL."
430 END

```

MATEMATICAS

Cambio de coordenadas cartesianas a esféricas y viceversa

En el espacio vectorial real un punto queda definido por tres coordenadas, una por cada dimensión espacial.

A cada punto le podemos hacer corresponder un vector que queda definido por el afijo, que es el punto anteriormente señalado. De esta forma un vector cualquiera V lo podemos expresar como $V(x,y,z)$, donde cada término representa lo que llamamos una coordenada en una base ortonormal. Veamos qué significa esto:

Base. Base de un espacio vectorial es un sistema de vectores que cumple las dos características siguientes:

- Son libres, es decir, que ninguno de ellos puede ponerse como la suma de los demás.

- Son un sistema generador, es decir, que cualquier vector del espacio puede ponerse como suma de los vectores de la base.

Base ortonormal: Una base ortonormal es aquella cuyos vectores cumplen:

- Son base.
- El módulo de los vectores es la unidad.
- El ángulo entre ellos es de 90° .

La representación de un vector (o de un punto, que no es más que el afijo del vector) en esta base es lo que denominamos representación en coordenadas cartesianas de un vector.

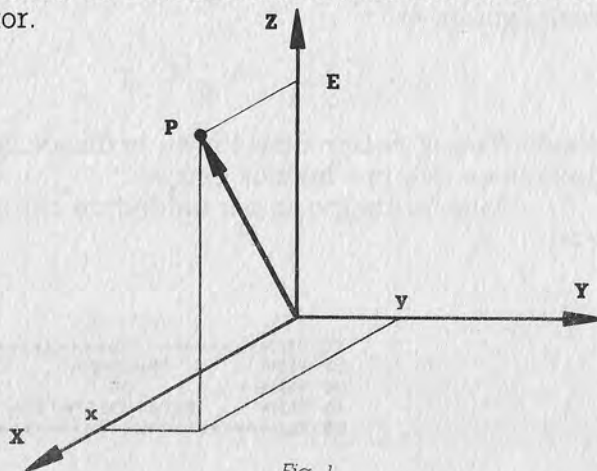


Fig. 1.

Para determinadas aplicaciones, sobre todo cuando trabajamos con cuerpos de geometría esférica o cilíndrica, la representación en coordenadas cartesianas resulta demasiado complicada por lo que necesitamos unas herramientas de cálculo más sencillas que las coordenadas cartesianas. Existen varios tipos de coordenadas distintas de las cartesianas, pero aquí nos vamos a centrar en la representación en coordenadas esféricas, por su importancia tanto en matemáticas (resolución de integrales triples, transformaciones...) como en física (cálculo de momentos de inercia, seguimiento aéreo, campos...).

Un vector en coordenadas esféricas queda definido por:

- Su módulo, que como el módulo de cualquier vector queda definido por longitud del vector.

- Ángulo TETHA, que es el ángulo que forma el vector con el eje -X-.

- Ángulo PHI, que es el ángulo que forma el vector con el eje -Z-.

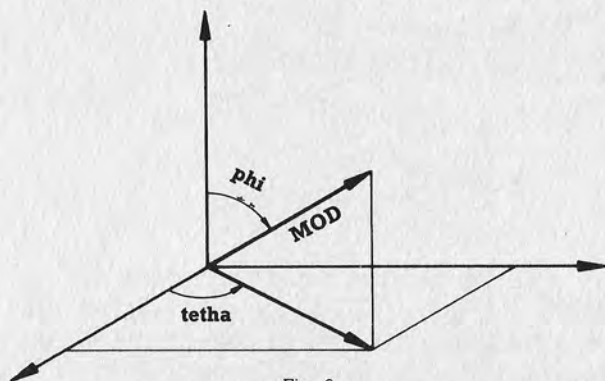


Fig. 2.

Entonces, si queremos calcular las coordenadas esféricas a partir de las coordenadas cartesianas, tendremos que realizar los siguientes cálculos, recordando que un vector se descompone en el plano de la siguiente forma:

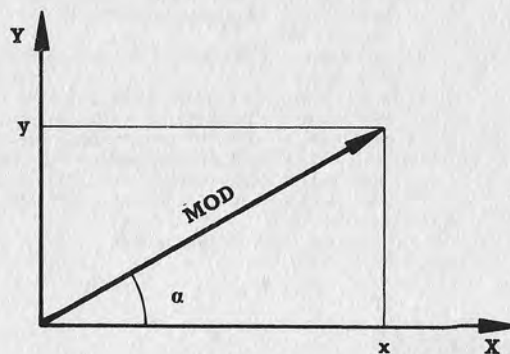


Fig. 3.

$$x = \text{MOD} \cdot \cos(\alpha)$$

$$y = \text{MOD} \cdot \sin(\alpha)$$

Luego, en el espacio, el vector queda expresado en esféricas como:

$$\text{MOD} = \sqrt{x^2 + y^2 + z^2}$$

$$\text{TETHA} = \text{ARCTAN}(y/x)$$

$$\text{PHI} = \text{ARCTAN}(\sqrt{x^2 + y^2}/z)$$

De forma análoga, si un vector me viene dado en esféricas y quiero pasarlo a cartesianas:

$$x = \text{MOD} \cdot \sin(\text{PHI}) \cdot \cos(\text{TETHA})$$

$$y = \text{MOD} \cdot \sin(\text{PHI}) \cdot \sin(\text{TETHA})$$

$$z = \text{MOD} \cdot \cos(\text{PHI})$$

```

10 REM*****
20 REM      PROGRAMA      *
30 REM      DE            *
40 REM      CAMBIO DE COORDENADAS *
50 REM*****
60 REM      VALIDO PARA AMSTRAD *
70 REM      IBM-PC Y COMPATIBLES *
80 REM      COMMODORE, SPECTRUM *
90 REM      Y MSX          *
100 REM*****
110 CLS
120 LET PI=3.141593
130 PRINT "ESTE PROGRAMA REALIZA EL CAMBIO"
140 PRINT "DE COORDENADAS ESFERICAS A CARTESIANAS"
150 PRINT "Y VICEVERSA"
160 PRINT
170 PRINT "1. CAMBIO DE CARTESIANAS A ESFERICAS"
180 PRINT "2. CAMBIO DE ESFERICAS A CARTESIANAS"
190 INPUT "INTRODUZCA OPCION";E
200 IF E=1 THEN GOSUB 230
210 IF E=2 THEN GOSUB 340
220 GOTO 430

```

```

230 REM***** AQUI CALCULO ESFERICAS*****
240 PRINT
250 INPUT "INTRODUCE COORDENADA -X-":X
260 INPUT "INTRODUCE COORDENADA -Y-":Y
270 INPUT "INTRODUCE COORDENADA -Z-":Z
280 PRINT "EL MODULO VALE :";SQR(X^2+Y^2+Z^2)
290 IF X=0 THEN PRINT "EL ANGULO TETHA VALE ";PI/2;" RADIANTES":GOTO 310
300 PRINT "EL ANGULO TETHA VALE: ";ATN(Y/X);" RADIANTES"
310 IF Z=0 THEN PRINT "EL ANGULO PHI VALE: ";PI/2;" RADIANTES":GOTO 330
320 PRINT "EL ANGULO PHI VALE: ";ATN(SQR(X^2+Y^2)/Z);" RADIANTES"
330 RETURN
340 REM ***** AQUI CALCULO CARTESIANAS*****
350 PRINT
360 INPUT "INTRODUCE EL MODULO ":MOD
370 INPUT "INTRODUCE ANGULO TETHA ":T
380 INPUT "INTRODUCE ANGULO PHI ":P
390 PRINT "LA COORDENADA -X- VALE :";MOD*SIN(T)*COS(P)
400 PRINT "LA COORDENADA -Y- VALE :";MOD*SIN(T)*SIN(P)
410 PRINT "LA COORDENADA -Z- VALE :";MOD*COS(T)
420 RETURN
430 REM *****FINAL*****

```

SOCIEDAD

Proposiciones

En el lenguaje cotidiano empleamos numerosas proposiciones simples de las cuales podemos afirmar, sin duda, que son verdaderas o falsas. Estas proposiciones simples las combinamos de muchas maneras, mediante conectivos o nexos, formandose así las proposiciones compuestas. Según el nexo o conectivo que se utilice, se forman los distintos tipos de proposiciones compuestas.

Entre estos tipos o conectivos que se utilizan encontramos:

- igual a b
- $\neg$:negación, "NO"
- $\vee$:disyunción, "O"
- $\wedge$:conjunción, "Y"
- $\Rightarrow$:implicación simple.
- $\Leftrightarrow$:implicación compuesta.

Tipos de proposiciones compuestas

1. *Conjunción de proposiciones.* Si dos proposiciones simples, p y q , las unimos mediante el conectivo "-Y-", se forma una tercera proposición (proposición compuesta), que se llama conjunción de p y q . Esta proposición compuesta es verdadera sólo cuando sean verdaderas las proposiciones p y q .

Tabla de verdad

P	Q	P-Y-Q
V	V	V
V	F	F
F	V	F
F	F	F

2. *Disyunción de proposiciones.* Al unir dos proposiciones simples p y q mediante el conectivo "-O-", se forma una nueva proposición compuesta llamada disyunción de proposiciones p y q . Se representa por $p \vee q$. La disyunción de dos proposiciones es falsa sólo y cuando las dos proposiciones p y q sean falsas.

Tabla de verdad

P	Q	P-O-Q
V	V	V
V	F	V
F	V	V
F	F	F

3. *Negación de proposiciones.* La negación $\neg p$ (o $\neg p$) de una proposición simple p consiste en negar lo que afirma la proposición simple p . Se representa por $\neg p$.

P	$\neg P$
V	F
F	V

4. *Proposición condicional.* Cuando dos proposiciones simples p y q las unimos mediante "si...entonces...", la proposición compuesta formada se llama condicional o implícita simple. Se representa por $p \Rightarrow q$.

Tabla de verdad

P	Q	P= $\Rightarrow$ Q
V	V	V
V	F	F
F	V	V
F	F	V

5. *Proposición recíproca.* La proposición condicional $p \Rightarrow q$ tiene una proposición recíproca: $q \Rightarrow p$.

6. *Proposición contraria.* La proposición condicional $p \Rightarrow q$ tiene la proposición condicional contraria $\neg p \Rightarrow \neg q$

7. *Proposición contrarrecíproca.* La proposición condicional $p \Rightarrow q$ tiene la proposición contrarrecíproca $\neg q \Rightarrow \neg p$.

8. *Proposiciones bicondicionales.* Si dos proposiciones simples se unen mediante

el conectivo "si y solo si...entonces..." se forma una proposición compuesta llamada bicondicional o de doble implicación.

P	Q	$P \Leftrightarrow Q$
V	V	V
V	F	F
F	V	F
F	F	V

```

10 REM*****
20 REM*      PROGRAMA      *
30 REM*      DE            *
40 REM*      PROPOSICIONES *
50 REM*****
60 REM*  VALIDO PARA AMSTRAD *
70 REM*  IBM-PC Y COMPATIBLES *
80 REM*  COMMODORE, SPECTRUM *
90 REM*      Y MSX         *
100 REM*****
110 PRINT "ESTE PROGRAMA LE MUESTRA VARIOS"
120 PRINT "EJEMPLOS DE PROPOSICIONES LOGICAS,"
130 PRINT "EL ESTUDIO SE CENTRA EN LA CONJUNCION,"
140 PRINT "DISYUNCION Y NEGACION DE PROPOSICIONES."
150 PRINT
160 PRINT "*****EJEMPLO DE CONJUNCION*****"
170 PRINT
180 PRINT "P=EL NUMERO 7 ES IMPAR:V"
190 PRINT "Q=EL NUMERO 7 ES NUMERO PRIMO:V"
200 PRINT "P-Y-Q=EL NUMERO 7 ES IMPAR Y NUMERO PRIMO:V"
210 PRINT
220 FOR F=1 TO 4
230 READ A
240 READ A$
250 PRINT "EL NUMERO ";A;"ES PAR:":INPUT V$
260 GOSUB 3400
270 READ A$
280 PRINT "EL NUMERO ";A;"ES DE DOS CIFRAS: ":INPUT V$
290 GOSUB 3400
300 READ A$
310 PRINT "EL NUMERO";A;"ES PAR Y DE DOS CIFRAS":INPUT V$
320 GOSUB 3400
325 PRINT
330 NEXT F
335 REM*****DISYUNCION DE PROPOSICIONES*****
336 PRINT
337 PRINT "***** EJEMPLO DE DISYUNCION *****"
340 FOR F=1 TO 4
350 READ A
360 READ A$
370 PRINT "EL NUMERO ";A;"ES PAR:":INPUT V$
380 GOSUB 3400
390 READ A$
400 PRINT "EL NUMERO ";A;"ES DE DOS CIFRAS: ":INPUT V$
410 GOSUB 3400
420 READ A$
430 PRINT "EL NUMERO";A;"ES PAR -O- DE DOS CIFRAS":INPUT V$
440 GOSUB 3400
450 NEXT F
500 PRINT
510 PRINT "HAS ACERTADO ";C;" VECES"
520 GOTO 4000
3400 REM*****SUBROUTINA DE DETENCION DE ERRORES*****
3410 IF A$(>)V$ THEN PRINT "LA CONTESTACION VALIDA ERA: ";A$
3510 IF A$=V$ THEN PRINT "CORRECTO ":C=C+1
3610 RETURN
3710 DATA 12,"V","V","V",8,"V","F","F",17,"F","V","F",3,"F","F","F"
3810 DATA 10,"V","V","V",3,"F","F","F",8,"V","F","V",13,"F","V","V"
4000 REM*****FINAL*****

```

APRENDER CON EL ORDENADOR

PARA LOS MAS JOVENES

La moneda

Todos los países del mundo poseen su propia moneda; en Inglaterra, la libra; en EE.UU., el dólar, etc. En España ya sabéis que es la peseta.

Con el paso del tiempo han ido apareciendo y desapareciendo en nuestro mercado monetario billetes y monedas. Ahora mismo en

España circulan billetes de 10.000, 5.000, 2.000, 1.000, 500 y 200 pesetas, y casi en desuso los billetes de 100 pesetas y monedas de 1, 2, 5, 25, 50, 100 y 200 pesetas.

Hemos hecho este programa para que te familiarices y manejes con mayor facilidad los billetes y monedas que existen en el mercado monetario de España. Queremos que aprendas a manejar correctamente los billetes y monedas dependiendo de la cantidad exigida.

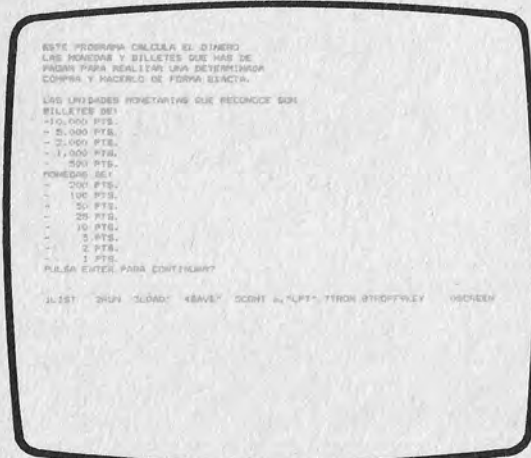


Fig. 4.



Fig. 5.

```

10 REM*****
20 REM*      PROGRAMA      *
30 REM*      DE            *
40 REM* DEVOLUCION DE MONEDAS *
50 REM*****
60 REM* VALIDO PARA AMSTRAD *
70 REM* IBM-PC Y COMPATIBLES *
80 REM* COMMODORE, SPECTRUM *
90 REM*      Y MSX        *
100 REM*****
110 CLS
120 PRINT "ESTE PROGRAMA CALCULA EL DINERO"
130 PRINT "LAS MONEDAS Y BILLETES QUE HAS DE"
140 PRINT "PAGAR PARA REALIZAR UNA DETERMINADA"
150 PRINT "COMPRA Y HACERLO DE FORMA EXACTA."
160 PRINT
170 PRINT "LAS UNIDADES MONETARIAS QUE RECONOCE SON"
180 PRINT "BILLETES DE:"
190 PRINT "-10.000 PTS."
200 PRINT "- 5.000 PTS."
210 PRINT "- 2.000 PTS."
220 PRINT "- 1.000 PTS."
230 PRINT "- 500 PTS."
240 PRINT "MONEDAS DE:"
250 PRINT "- 200 PTS."
260 PRINT "- 100 PTS."
270 PRINT "- 50 PTS."
280 PRINT "- 25 PTS."
290 PRINT "- 10 PTS."
300 PRINT "- 5 PTS."
310 PRINT "- 2 PTS."
320 PRINT "- 1 PTS."
330 INPUT "PULSA ENTER PARA CONTINUAR";A$
340 CLS
350 INPUT "INTRODUCE EL IMPORTE DE LA COMPRA";C
360 PRINT

```

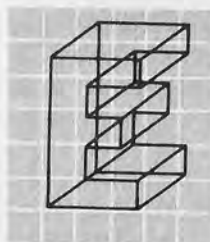
```

361 LET P=C
370 READ B:READ A$
380 LET H=0
400 LET H=INT(P/B)
420 PRINT "CUANTOS ";A$;" NECESITO ? ":INPUT V
430 IF V<>H THEN PRINT "NO ES CORRECTO, NECESITAS ";H;" ";A$
440 LET P=P-H*B
450 IF P=0 THEN END
455 GOTO 370
460 DATA 10000,BILLETES DE 10000,5000,BILLETES DE 5000,2000,BILLETES DE 2000
470 DATA 1000,BILLETES DE 1000,500,BILLETES DE 500,200,MONEDAS DE 200,100
480 DATA MONEDAS DE 100,50,MONEDAS DE 50,25,MONEDAS DE 25,10,MONEDAS DE 10
490 DATA 5,MONEDAS DE 5,2,MONEDAS DE 2,1,MONEDAS DE 1

```


PEQUEÑA HISTORIA DE LA INFORMATICA

■ Investigación con fines militares: el proyecto Sage



N otros momentos hemos hablado de Alan Turing, personaje en extremo excéntrico, que ayudó tanto a los Servicios de Información Británicos en la tarea de descifrar mensajes alemanes con el ordenador Colossus, instalado en Bletchley Park, cerca de Londres. El Colossus, llegó a ser la «media naranja» del Enigma, la máquina de cifrado alemana. Esta última máquina había sido fabricada con enormes medidas de seguridad por los alemanes, y utilizaba claves con millones de permutaciones, por lo que parecía absolutamente imposible descifrar mensaje alguno. Pero la resistencia polaca consiguió obtener un ejemplar de la máquina, que fue enviada a Londres, y tras su estudio, se llegó a poder descifrar los mensajes, con ayuda de un ordenador: el Colossus del que antes hemos hablado. De este ordenador llegaron a fabricarse diez ejemplares. El que fue instalado en Bletchley Park, funcionaba bajo la dirección de Alan Turing, cuyo trato humano desde luego no podía compararse al de Von Neumann, mucho más agradable y campechano.

Acabada la guerra, y satisfechos los británicos con el resultado de los Colossus, a pesar de que eran máquinas de tubos (tenían unos dos mil quinientos y podían procesar unos 25.000 caracteres por segundo), el Gobierno Británico consideró la posibilidad de desarrollar un proyecto propio de bomba atómica. Y para este proyecto podía ser útil contar con un ordenador británico cien por cien.

El Departamento de Defensa decidió encargar el diseño a unos fabricantes de armas (también introducidos en el mundo de la electrónica) a los que conocía bien como proveedores. Ferranti Ltd. de Manchester comenzó a desarrollar el proyecto en 1949 basándose en el prototipo del MARK I inglés, que había funcionado muy bien, y tenía el honor de ser uno de los primeros ordenadores a nivel mundial.

En 1951, el proyecto había sido terminado, y la Ferranti consiguió vender varias máquinas, aprovechando el ser proveedor del Gobierno en el British Atomic Weapons Research Establishment-Centro Británico de Desarrollo de Armamento, y algunas otras a otras entidades gubernamentales. Con la idiosincrasia inglesa, el ser un fabricante inglés, y además proveedor de ordenadores para el Ejército, le abrió otras muchas puertas, a nivel institucional. Es curioso constatar que por el momento, existía la creencia generalizada que esas máquinas enormes y sofisticadísimas no tendrían utilidad fuera de ciertas tareas muy específicas y para entidades cuyas finanzas fueran más que potentes. No era esta idea propia únicamente del gran público, no muy versado por lo general en temas técnicos, también era sostenida por personas que colaboraron en el desarrollo de proyectos muy interesantes. Por ejemplo, Howard Aiken, en Estados Unidos, siempre pensó que Eckert y Mauchly se estaban equivocando de medio a medio en el desarrollo del Univac I, ya que jamás tendrían clientes suficientes para hacer viable la Compañía.

Como se ha visto con el tiempo, la predicción de acontecimientos está más unida al saber popular antiguo, al conocimiento de las realidades cotidianas que a las inteligencias

PEQUEÑA HISTORIA DE LA INFORMATICA

de primer orden. Además, otros descubrimientos ayudarían a dar la razón al dúo Eckert-Mauchly.

Volviendo al Colossus inglés, los Ferranti consiguieron vender una de sus máquinas a la Universidad de Toronto, en Canadá, para uno de sus Laboratorios de Investigación, en concreto el de Fluidos.

Sin embargo, la tónica general de la época era utilizar poco los enormes y delicadísimos ordenadores para tareas civiles, limitándolos generalmente en empresas muy poderosas a trabajos comunes de oficina, que cubrían perfectamente. Apenas parecía que hubiera otras necesidades, y probablemente los ordenadores no se habrían desarrollado como lo han hecho, si las demandas de características superiores por parte del ejército no los empujaban a cotas muy superiores. Del proyecto militar ambicioso SAGE, surgieron otros proyectos civiles, como el SABRE en 1958, que cubría todas las operaciones de reservas de plazas de avión (las compañías de aeronáutica estaban sufriendo un desarrollo enorme también en estos años). La Lockheed, por ejemplo, ya por esas fechas disponía de un sistema de seguimiento de la fabricación de sus aviones mediante ordenador. No hay que olvidar que la Lockheed trabajaba mucho con la Aviación norteamericana, y era fundamental para ella ofrecer un buen servicio, aunque los precios subieran «algo más» de lo previsto en un principio.

Pasemos a hablar del importantísimo proyecto SAGE desarrollado en la mitad de nuestro siglo.

La Aviación norteamericana se dirigió al Massachusetts Institute of Technology (MIT) para desarrollar un estudio de un sistema de defensa, basado en tiempo real. El país se encontraba en plena guerra fría con la URSS, y el proyecto era importante. A MIT no le faltaban recursos ni económicos ni humanos para sacar adelante el proyecto.

El prototipo a utilizar sería el Whirlwind, pero, naturalmente, mejorado, sobre todo en lo concerniente a sus averías. El Organismo creado para controlar el proyecto fue el Air Defense System Engineering Committee, que era el que había propuesto el proyecto de defensa SAGE (Semi Automatic Ground Environment) de defensa antiaérea automática, utilizando ordenadores. El sistema de defensa consistía básicamente en radares, cuya información pasaba automáticamente a los ordenadores. Estos, localizado el objetivo y el

caza correspondiente, dirigían el proyectil desde el caza al objetivo, neutralizándolo. En abril de 1951 se realizaron las primeras pruebas que tuvieron un considerable éxito. En la pantalla, aparecían objetivo y caza, y el ordenador iba calculando las trayectorias en tiempo real, para que el caza interceptara siempre al objetivo con sus proyectiles, aunque éste estuviera cambiando rumbo, altitud, etc., constantemente.

Este proyecto supuso una verdadera revolución. A partir de él se desarrollaron muchísimos otros de uso civil, todos ellos de unas características de muy alto nivel. En el proyecto SAGE, cada ordenador estaba unido a varias consolas de visualización, donde trabajaban los operadores (primera operación a tiempo compartido). Los aviones eran «reconocidos» comparando distintos planos almacenados en la memoria del ordenador con el del objetivo. Esta última característica supone una de las bases para los simuladores de vuelo. El proyecto contaba con otras muchas características que serían fundamentales en la Historia de la Informática.

Por razones de seguridad, cada centro disponía de un sistema de dos ordenadores idénticos, de forma que si uno de ellos fallaba, el otro pasaría a funcionar instantáneamente. La red estaba formada por veintitrés sectores aéreos que cubrían las necesidades de defensa, tres centros de coordinación y un centro de investigación (el Lincoln Laboratory).

En 1952, el Lincoln Laboratory se dirigió a IBM para que colaborase en el proyecto SAGE. La empresa debería suministrar cincuenta ordenadores.

El primer centro SAGE fue operativo en el año 1958 ubicando la base del proyecto en Mac Guire, New Jersey. Estos centros eran casi autónomos; disponían entre otros elementos de generadores eléctricos propios, punto fundamental, ya que el consumo eléctrico de las máquinas, más el debido al aire acondicionado, era altísimo. No debemos olvidar que estos grandes ordenadores llevaban más de 60.000 tubos de vacío, que consumían una buena parte de los 750 kW hora, que era el consumo total del equipo. Eran máquinas muy potentes, que podían controlar hasta 400 aviones, utilizando una centena de radares. Su peso (275 toneladas) daba una idea de su tamaño. Se realizaron múltiples simulaciones para «preparar al personal», aunque afortunadamente nunca fue utilizado realmente por algún

ataque enemigo. La fiabilidad de estos ordenadores fue quizá su mayor ventaja, siendo de cinco o seis horas anuales, realmente increíble para la época.

El sistema SAGE fue perfeccionándose, y ha sido uno de los grandes acicates en la industria informática. A IBM, este proyecto le supone un gran desarrollo, pero no fue la única casa fabricante en beneficiarse. Las primicias de casi todos los descubrimientos en el campo de los ordenadores han ido aplicándose al sistema SAGE, mejorándolo y perfeccionándolo poco a poco, hasta un cierto momento, en el que mejores sistemas lo desbancaron. Los circuitos integrados revolucionaron los equipos y proyectos anteriores, y fue necesario partir de consideraciones distintas. En 1984 se cerró el último centro SAGE.

El sistema SAGE fue probablemente el sistema más importante de defensa automática. Sin embargo, existieron otros proyectos, también muy interesantes. El NORC (Naval Ordnance Research Calculator) fue un ordenador creado especialmente para la Marina por IBM. Data de 1954, y Von Neumann colaboró también en el proyecto. Ya hemos hablado del 701 y del 704 de IBM. De este último, desciende el 709, también pensado con fines militares.

Entre otras empresas que colaboraron en proyectos militares para el Gobierno americano cabe citar a ERA (Electronic Research Company), que en cierto sentido descendía de UNIVAC, y que en 1953 diseñó el primer ordenador de tambor, el 1101 para la Georgia TECH; la propia UNIVAC, que construyó el M460, RCA, que diseñó el BIZMAC.

La tendencia del mercado de ordenadores, tan influido por las necesidades de los distintos sistemas defensivos, era incrementar sobre todo la velocidad. Los ordenadores convencionales trabajaban a velocidades cada vez más rápidas, pero aquellos que eran diseñados para usos científicos o militares necesitaban mejorar mucho más las tasas de velocidad. Una gran parte de las tareas a realizar necesitaban trabajar en tiempo real, y naturalmente, en estos casos la velocidad es primordial.

Si tomamos como medida de la velocidad de cada ordenador el tiempo que tarda

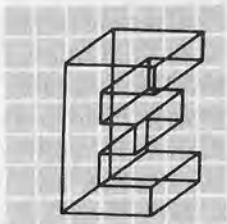
en efectuar una suma, vemos que durante la primera generación de ordenadores los primeros UNIVAC realizaban unas 2.000 sumas por segundo (500 microsegundos por suma), llegando el UNIVAC III a efectuar unas 33.000 sumas por segundo. En cuanto a las multiplicaciones, el MARK I necesitaba tres segundos para multiplicar números de diez dígitos, el IBM 650, unos 2 milisegundos, el IBM 701 450 microsegundos. El ERA, del que acabamos de hablar utilizaba unos 260 microsegundos, y el NORK unos 30. Con estos datos, hemos podido ver que en diez años la velocidad de un ordenador se había incrementado por 100.000.

Al final de la llamada «primera generación», se había decantado la necesidad de fabricar ordenadores, y los calculadores y otras máquinas de uso limitado se iban fabricando cada día menos. Estos calculadores han vuelto a fabricarse «con otro envoltorio distinto» con las calculadoras de bolsillo. Naturalmente, para salvar la distancia entre un calculador enorme y una pequeña calculadora han sido necesarios algunos años, y un progreso tecnológico sin precedentes, pero estas pequeñas máquinas, que hoy tiene casi cualquiera, realizan las mismas operaciones que los grandes calculadores de hace unos años, a una velocidad mayor, y, todo ello, sin considerar su tamaño.

También, con el final de la primera generación de ordenadores, las Universidades fueron perdiendo poco a poco su papel preponderante en su desarrollo. Naturalmente, en nuestros días siguen existiendo laboratorios de desarrollo en todas las Universidades, las americanas e inglesas, que fueron las pioneras, y en todas las del mundo. Sin embargo, los campos se han diversificado. Las Universidades siguen investigando, mejorando, asesorando, pero carecen de los recursos e infraestructura necesarias para fabricar ordenadores. A partir de esta primera generación, el mercado ha ido pasando a las Compañías especializadas que se han ido creando, y que han seguido invirtiendo e investigando en todos los campos relacionados con los ordenadores. Ya no se fabrican prototipos, la segunda generación de ordenadores se basa en máquinas fabricadas en serie, para un mercado cada vez más amplio y selectivo.

PROCESOS INFERENCIALES EN LOS SISTEMAS BASADOS EN EL CONOCIMIENTO (I)

Reglas de producción



El formalismo de la lógica que hemos comentado permite la definición cómoda y precisa de ciertas propiedades que pueden cumplir los elementos de la base de conocimientos o de algunas relaciones que se pueden

dar entre ellos. Por ejemplo, podemos establecer la norma siguiente (un tanto simplista) de análisis zoológico: "SI un animal es ave, tiene dos patas y es de color blanco o negro ENTONCES es un avestruz". Esta expresión no es más que una proposición con una conectiva "implicación", cuatro antecedentes ("es ave", "tiene dos patas", "es blanco", "es negro") y una conclusión ("es avestruz").

Esta proposición representa una información de tipo deductivo: permite obtener nuevas informaciones a partir de las ya existentes; es decir, si ya sabemos que el animal objeto de nuestro análisis "es ave y tiene dos patas y es o bien blanco o bien negro", la proposición que comentamos nos indica que podemos concluir que "es un avestruz". Cuando se manejan estos conceptos como tales expresiones lógicas, en vez de la proposición anterior con una implicación se suele formular el mismo contenido de información con un símbolo especial " $\Rightarrow$ " que se lee "permite deducir". Se escribe "el animal es ave" y "tiene dos patas" y ("es blanco" o "negro") $\Rightarrow$ "es un avestruz".

Estas expresiones que se adecúan a la estructura formal "SI condición1 $\wedge$ condición2 $\wedge$ condición3, etc., ENTONCES acción1 $\wedge$ acción2 $\wedge$ acción3, etc.", donde $\wedge$ puede ser una conjunción ("Y", $\wedge$) o una disyunción ("O", $\vee$), se llaman "reglas de producción", debido al carácter deductivo que comportan, que permite

"producir" nuevas informaciones a partir de las ya disponibles. Las "condiciones" que aparecen en la primera parte de la regla se suelen llamar también "premisas" o "cláusulas SI"; las acciones que aparecen a la derecha del ENTONCES (que se corresponde con el símbolo de implicación, " $\Rightarrow$ ") se llaman "conclusiones" o "cláusulas ENTONCES". En el antecedente de la regla, las premisas deben ir unidas por la conectiva "Y" (aunque alguna de estas premisas sea una condición compuesta de otras varias mediante la disyunción "O"). El consecuente de la regla suele tener, en la mayoría de los casos, una sola conclusión.

Aunque el formalismo de las reglas de producción se corresponde con el general de la lógica, este tipo de expresiones se utilizan no sólo en los sistemas que usan como esquema de representación del conocimiento la lógica, sino también en aquéllos que se basan en representaciones de tipo red semántica o inferencial, ternas objeto-atributo-valor, marcos, etc. Así, por ejemplo, en un sistema experto famoso (el sistema MYCIN, diseñado para el diagnóstico de enfermedades infecciosas) que utiliza para la representación del conocimiento el esquema de ternas O-A-V, la mayoría del conocimiento del experto está expresado en forma de "reglas de producción" como la siguiente.

SI la base del cultivo es sangre y la morfología del organismo es bastón y la reacción gram es gramneg, ENTONCES hay una probabilidad grande (0,6) de que la identidad del organismo sea pseudomonas-aeruginosa.

Como se ve, en esta regla aparecen tres premisas y una conclusión y, en efecto, la estructura de los cuatro elementos es la de ob-

TEMAS MONOGRAFICOS DE VANGUARDIA

jeto-artículo-valor (cultivo-base-sangre; organismo-morfología-bastón; organismo-reacción gram-gramneg; organismo-identidad-seudomonas aeruginosa).

En la regla presentada, por otro lado, aparece un elemento que suele incluirse con frecuencia en las reglas de producción: los factores de incertidumbre que permiten realizar "razonamiento impreciso". La regla dada no se cumple siempre, indefectiblemente (en opinión del experto —el médico—), sino "con mucha frecuencia": por ello asigna una probabilidad relativamente grande (0,6 sobre 1) al hecho de que sea cierto que la identidad del organismo sea pseudomonas-aeruginosa, si se cumplen las tres premisas dadas.

Naturalmente, estas reglas de producción pueden incluir expresiones con variables (utilizando el formalismo de la lógica de primer orden). Por ejemplo, en vez de incluir las reglas:

regla 1: SI Adolfo es padre de Juan
 Juan es padre de Luis
 ENTONCES Adolfo es abuelo de Luis

regla 2: SI Ramiro es padre de Jonás
 Jonás es padre de Ramón
 ENTONCES Ramiro es abuelo de Ramón

regla 3: SI José es padre de Pedro
 Pedro es padre de Paco
 ENTONCES José es abuelo de Paco
etcétera.

Se pondría la regla genérica:

regla 4: SI X es padre de Y
 Y es padre de Z
 ENTONCES X es abuelo de Z

para, a partir de la base de hechos,

Adolfo es padre de Juan
Jonás es padre de Ramón
José es padre de Pedro, etc.

deducir que

Adolfo es abuelo de Luis
Ramiro es abuelo de Ramón,
etcétera.

«instanciando» o particularizando la regla 4 para los valores posibles que cumplan la condición clave de la regla: que el elemento de la derecha de la primera condición (Y) coincida con el elemento de la izquierda de la segunda condición. (Esta es, precisamente, la tarea a realizar por el motor de inferencia —no

sólo en este caso elemental, sino sobre todo en otros más sofisticados—, como veremos a continuación).

Los problemas a abordar en el diseño e implementación de un sistema basado en el conocimiento se resumen en tres grandes áreas: problemas de representación del conocimiento (adecuación del esquema elegido a la estructura formal de la información a representar, optimización de los algoritmos de captura del conocimiento, relación del formalismo de representación de datos con los algoritmos o técnicas de procesamiento de la información, etc.); problemas de inferencia del conocimiento (técnicas de procesamiento, estrategias de búsqueda, control de los procesos inferenciales, etc.); dificultades de relación con el usuario, con el creador de la base de datos, con el experto que facilita el conocimiento... (módulos de obtención del conocimiento, módulos de traza, mantenimiento de archivos históricos, módulos de explicaciones, etc.).

El primer grupo indicado se resuelve mediante una adecuada implementación de la base de conocimientos (tema del que ya hemos hablado); el segundo grupo de tareas son abordadas por el motor de inferencia del sistema; para las relaciones al exterior del conjunto se suelen diseñar módulos específicos, aunque, en ocasiones, se incluyen en el propio motor, instrumentos de control del proceso y de explicación y seguimiento de su actividad.

El motor de inferencia de un sistema experto es un conjunto de módulos que realizan las tareas básicas de proceso del conocimiento. Existe todo un cúmulo de actividades que debe realizar el Motor de Inferencia (MI) del sistema: proceso de deducción propiamente dicho, resolución de conflictos, gestión de prioridades, control del orden de utilización de las reglas, manejo de las memorias auxiliares (donde se van manteniendo los nuevos hechos deducidos, el estado del sistema y otras informaciones temporales, etc.), cálculo de los factores de incertidumbre «propagados» a través del sistema, etc.

El núcleo del MI es el módulo de deducción. Dependiendo del esquema utilizado en la representación del conocimiento, la estrategia de deducción puede variar. Sin embargo, la mayoría de los sistemas utilizan reglas de producción y su manejo se realiza mediante algunas (en numerosos casos una sola) reglas deductivas de la lógica. La estrategia básica utilizada es la regla llamada de «modus

ponens». En los sistemas que utilizan lógica de primer orden (lógica de predicados) se utilizan, en ocasiones otras «reglas» como «silogismo» o «modus tollens», según veremos más adelante.

La regla de «modus ponens» afirma que si es cierto (se puede afirmar, consta) que se cumple un hecho «A» y tenemos una regla de producción que asevera que « $A \rightarrow B$ » («de A se deduce B» o «A implica B» o «SI A ENTONCES B»), seguro que se puede afirmar el hecho B. Esto permite, a partir de los hechos que se conocen cuando comienza el proceso de deducción, obtener nuevos hechos. Naturalmente, si disponemos de otra regla que afirme « $B \rightarrow C$ », podemos aplicar de nuevo «modus ponens» para añadir un nuevo hecho demostrado (C) en nuestra base de hechos.

La regla de «modus ponens» es sumamente simple y produce razonamientos sencillos y suficientemente potentes (en multitud de casos). Sin embargo, hay procesos deductivos, claros e intuitivos que no pueden hacerse (al menos de un modo directo) con esta regla.

Observemos, por ejemplo, el razonamiento: «Si el animal considerado es un avestruz, entonces es un ave» (ES AVESTRUZ $\rightarrow$ ES-AVE); «el animal que estamos considerando no es un ave» (NO ES-AVE); por tanto «dicho animal no es un avestruz» (NO ES-AVESTRUZ). Parece bastante claro que es correcto (y es muy simple); sin embargo no puede ser resuelto directamente por «modus ponens»; habría que «deducir» por algún procedimiento (existe un teorema en lógica que muestra que este paso es correcto) la expresión «si no es ave no puede ser avestruz» a partir de la frase dada (es decir: obtener NO ES-AVE $\rightarrow$ NO ES-AVESTRUZ a partir de ES-AVESTRUZ $\rightarrow$ ES-AVE) y aplicar después «modus ponens» con «NO ES-AVE $\rightarrow$ NO ES-AVESTRUZ» y «NO ES-AVE» para obtener «NO ES-AVESTRUZ».

Existen otras transformaciones básicas que estudia la LÓGICA y son de aplicación en el procesamiento de expresiones de este tipo, como la equivalencia de «ES AVESTRUZ $\rightarrow$ ES-AVE» con «O NO ES-AVESTRUZ O ES-AVE» y con «no puede ser que se cumpla simultáneamente que ES-AVESTRUZ y NO ES-AVE», pero en la mayoría de los casos se utiliza exclusivamente la regla de «modus ponens» para producir las sucesivas deducciones.

Sin embargo, la aplicación de esta regla deductiva no se puede realizar de un modo anárquico con todas las reglas de producción que existen en la base de conocimien-

tos y con todos los hechos que estén demostrados en la base de hechos y/o memorias auxiliares. Deben definirse de algún modo las estrategias de búsqueda para que el proceso sea más eficaz.

Control del proceso deductivo

Aunque, como se ha visto, el proceso de obtención de una conclusión (mediante aplicación de la regla general de deducción, «modus ponens», a un hecho conocido) es un paso sumamente simple, el proceso general deductivo es más complejo. Hay dos problemas básicos en el funcionamiento del motor de inferencia:

a) ante todo, cómo comenzar el proceso y/o cuál es el objetivo a alcanzar: en efecto, la base de conocimientos es un cúmulo de información estática y habrá que arrancar el proceso de algún modo. Pueden utilizarse procedimientos diferentes: en ocasiones se marca(n) alguno(s) hecho(s) básico(s) con los que el motor debe comenzar su actividad deductiva; en otros casos se marca un objetivo a obtener (un hecho a demostrar) y es tarea del sistema ver si es demostrable o no; hay sistemas que admiten la presentación de un objetivo genérico (un conjunto de varios hechos que pueden ser demostrados) y el sistema elabora un proceso deductivo para ver, a partir de los hechos ya conocidos, cuál es el que se puede demostrar... ¡veremos ésto en detalle a continuación!

b) por otro lado, el motor de inferencia debe resolver los conflictos que se presentan cuando, en su tarea deductiva, puede elegir varias líneas de razonamiento alternativas (a partir de diferentes reglas que pueden ser aplicables). Debe disponer de «normas» de funcionamiento o «reglas sobre cómo aplicar las reglas»; o bien, debe tener capacidad para consultar al usuario... o ambas cosas a la vez.

Se puede hablar de unas técnicas generales de conducción del proceso deductivo: encadenamiento adelante o atrás, búsqueda en profundidad o en anchura, utilización de reglas de tipo «ver primero» o heurísticas (control por el contenido de las reglas), etc., aunque existen numerosos procedimientos de control adicionales que se suelen utilizar en unos u otros de los sistemas expertos y/o herramientas disponibles (activación y desactivación de paquetes de reglas o control por «objetivos ficticios» y «datos ficticios» en Intelligence Service; «gestión de hipótesis» y utili-

TEMAS MONOGRAFICOS DE VANGUARDIA

zación de primitivas de metaconocimiento es SNARK; control de reglas que se activan pasado un umbral de certeza y reglas de uso exclusivamente en encadenamiento adelante en M.I.M.I., etc.).

Veremos en detalle las técnicas generales citadas anteriormente y comentaremos los restantes procedimientos de control para que el lector tenga una visión lo más completa posible de las técnicas utilizadas para la gestión del proceso deductivo en los sistemas basados en el conocimiento.

■ Encadenamiento adelante y atrás

Si arrancamos el proceso deductivo a partir de un conjunto de reglas y uno o varios hechos conocidos, se puede aplicar la regla de «modus ponens» para ir obteniendo hechos nuevos demostrados. Así se va «avanzando» a través de la base de conocimientos obteniendo sucesivamente nuevos hechos a utilizar en los pasos posteriores: esto es el encadenamiento hacia adelante.

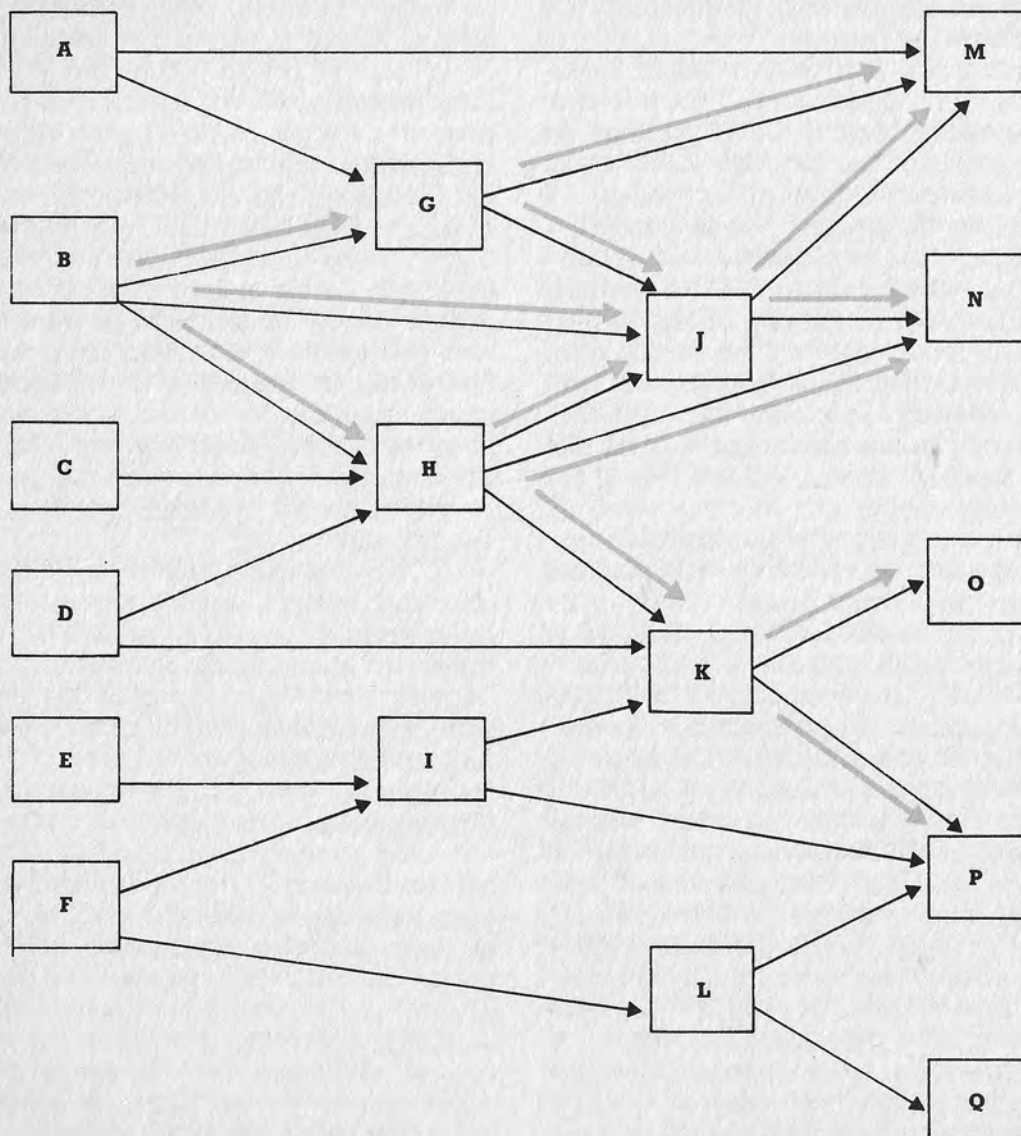


Fig. 1. Proceso de encadenamiento hacia adelante a partir de un hecho (B) avanzando por la base de conocimientos.

Si, por el contrario, queremos llegar a una conclusión (objetivo a obtener), se puede examinar qué reglas de producción tienen en su conclusión ese objetivo. Los antecedentes de la o las reglas que cumplen esa condición son los nuevos objetivos (subobjetivos se llaman a veces) a obtener, pues si podemos demostrar estos subobjetivos, automáticamente la aplicación de la regla correspondiente permitirá obtener el objetivo buscado. Con cada

uno de los subobjetivos se realiza la misma operación buscando las reglas en que son conclusión: los antecedentes (parte de condición o premisas) de esas reglas darán los nuevos subobjetivos a obtener... y por este procedimiento vamos «retrocediendo» por la base de conocimientos hasta los primeros hechos que conocemos o sobre los que se pregunta al usuario. Este es el encadenamiento hacia atrás.

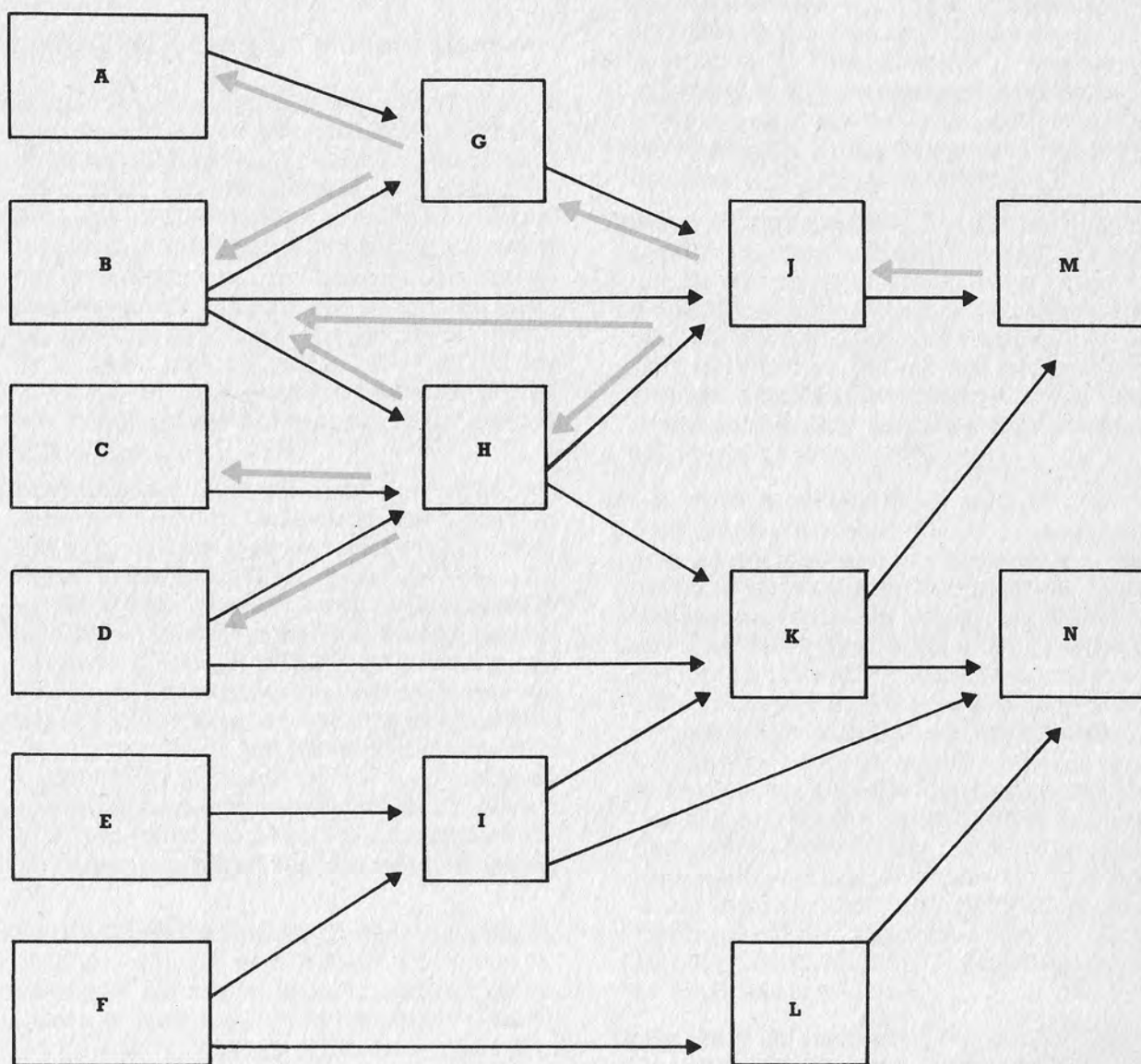


Fig. 2. Proceso de encadenamiento hacia atrás a partir del objetivo M retrocediendo hacia los hechos de base A, B, C,...

TERMINOLOGIA

Back chaining. Véase encadenamiento hacia atrás.

Backtracking dirigido a dependencias.

Técnica de programación que permite a un sistema eliminar los efectos de suposiciones incorrectas durante la búsqueda de la solución a un problema. Cuando el sistema infiere nueva información, mantiene los registros de dependencia de todas sus deducciones y suposiciones, mostrando cómo fueron derivadas. En el momento en que encuentra que una suposición fue incorrecta, realiza un backtracking (salto hacia atrás) en las cadenas de inferencias, eliminando aquellas conclusiones basadas en la suposición incorrecta.

Backtracking. Técnica de búsqueda que va seleccionando en cada nodo del gráfico el nodo adyacente que se tomará a continuación. Si este es un nodo que no satisface las condiciones que se están comprobando —«fallo»—, se tomará otro nodo adyacente al original. Si todos los nodos adyacentes son «fallos», el propio nodo considerado será «fallo». De esta forma, a través de la secuencia de inferencias se van probando los distintos caminos. En la planificación de problemas, este sistema se utiliza para ir probando uno tras otro todos los planes, de forma que vayan identificándose los que sean inaceptables.

Búsqueda en anchura. En una jerarquía de reglas u objetos, este tipo de búsqueda indica una estrategia en la que todas las reglas u objetos dentro del mismo nivel de jerarquía se examinan antes que cualquiera de las reglas y objetos del siguiente nivel inferior. En contraposición a búsqueda en profundidad.

Búsqueda en profundidad. En un sistema jerárquico de reglas u objetos, la búsqueda

en profundidad indica una estrategia en la cual se examina una regla y objeto del nivel más alto y a continuación las reglas u objetos situados inmediatamente debajo del examinado. Al proceder de este modo, el sistema busca por una rama simple del árbol jerárquico, hasta terminarla. En contraposición a búsqueda en anchura.

Búsqueda exhaustiva. Dícese de una búsqueda que es exhaustiva si se examina cada una de las ramas del árbol de decisión o red de decisión. (Este tipo de búsqueda es prácticamente imposible y muy costosa, por multitud de razones, en algunos casos concretos). Los sistemas basados en el conocimiento suelen realizar búsquedas exhaustivas en sus bases de conocimiento.

De arriba a abajo. Estrategia de procedimiento por la cual se va de lo complejo y abstracto a lo sencillo y concreto. Si se aplica como estrategia en la resolución de problemas, se refiere al método de comenzar con un problema complejo e ir descomponiéndolo en subproblemas que son más sencillos de resolver. Los sistemas con encadenamiento hacia atrás, suelen resolver los problemas por éste procedimiento. Si se aplica a la metodología de programación, el término se refiere a un estilo en el cual la organización abstracta del programa se va determinando en etapas en las que va aumentando la concreción, culminando por escribir los elementos primitivos que se pueden ejecutar directamente. La estrategia opuesta es de abajo a arriba.

Dirigido a un modelo. Otra forma de designar el encadenamiento hacia atrás. En contraposición a controlado o dirigido por los datos.

Dirigido hacia objetivos. Término que significa lo mismo que el Encadenamiento ha-

TERMINOLOGIA

cia atrás. Se utiliza en contraposición a dirigido a datos.

Dirigido por datos. Controlado por los cambios efectuados en los datos, en lugar de hacerlo por los cambios de objetivo. En contraposición a dirigido por objetivos. Véase demonio y encadenamiento hacia adelante.

Dirigido o controlado por un patrón. Controlado por ciertas configuraciones de datos. Los sistemas de producción son un caso especial de sistemas controlados por patrones.

Encadenamiento hacia adelante. Método de resolución de problemas, que comienza con un conocimiento inicial y aplica reglas de inferencia para generar nuevos conocimientos hasta que cualquiera de las inferencias satisface el objetivo, o no se pueden realizar más inferencias. En los sistemas de producción de encadenamiento hacia adelante, la aplicabilidad de una regla viene determinada por la satisfacción de las condiciones especificadas en su lado izquierdo, respecto del conocimiento almacenado actualmente en la memoria de datos.

Encadenamiento hacia atrás. Método de resolución de problemas. Es una de las distintas estrategias de control que regulan el orden en el que se establecen las inferencias. El sistema comienza marcando un objetivo a cumplir, y se va ampliando cada objetivo sin resolver en otros subobjetivos hasta encontrar la solución, o hasta que cada objetivo ha sido dividido en sus componentes más sencillos. Cuando se resuelve un subobjetivo, se pasa al objetivo padre. En los sistemas de producción de arquitectura de encadenamiento hacia atrás, la aplicación de una regla se determina examinando sus conclusiones (en el lado derecho) en lugar de sus condiciones antecedentes (en el lado izquierdo).

Estrategia de control. Método para selec-

cionar la acción siguiente entre varias fases alternativas de resolución de un problema. En sistemas de producción, el encadenamiento hacia atrás, es un ejemplo de estrategia de control.

Inferencias controladas por los datos. Inferencias controladas por hechos, en lugar de serlo por objetivos. Véase Encadenamiento hacia adelante.

Inferencias controladas por objetivos. Inferencias controladas por objetivos, en lugar de serlo por los datos. Véase Encadenamiento hacia atrás.

Método de inferencia. Técnica utilizada por el motor de inferencia para acceder y aplicar el conocimiento del dominio, por ejemplo, los encadenamientos hacia adelante y hacia atrás.

Métodos de base lógica. Métodos de programación que utilizan cálculos de predicados para estructurar el programa y guiar su ejecución.

Modus ponens. Regla de inferencia de la lógica: $A, A \rightarrow B \Rightarrow B$.

Motor de inferencia. Parte del sistema basado en el conocimiento o sistema experto que contiene el conocimiento para la resolución general del problema. El motor de inferencia procesa el conocimiento de un dominio (localizado en la base de conocimientos) para obtener nuevas conclusiones.

Subobjetivo. Objetivo que cuando se satisface es suficiente para asegurar que se ha obtenido también otro objetivo. En sistemas con encadenamiento hacia atrás, los objetivos sin alcanzar se descomponen en subobjetivos más simples, con la esperanza de que éstos se puedan resolver con mayor facilidad. En los árboles de los objetivos, la relación de un objetivo con sus subobjetivos está representada como una relación padre-hijo.

